

I2RS working group  
Internet-Draft  
Intended status: Informational  
Expires: September 13, 2017

S. Hares  
Huawei  
A. Dass  
Ericsson  
March 12, 2017

NETCONF Changes to Support I2RS Protocol  
draft-hares-netconf-i2rs-netconf-01.txt

Abstract

This document describes a NETCONF capability to support the Interface to Routing system (I2RS) protocol requirements for I2RS protocol version 1. The I2RS protocol is a re-use higher layer protocol which defines extensions to other protocols (NETCONF and RESTCONF) and extensions to the Yang Data Modeling language.

The I2RS protocol supports ephemeral state datastores as control plane datastores. Initial versions of this document contain descriptions of the ephemeral datastore. Future versions may move this description to NETMOD datastore description documents.

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 <http://datatracker.ietf.org/drafts/current/>.

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

This Internet-Draft will expire on September 13, 2017.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. Definitions Related to Ephemeral Configuration . . . . .	3
2.1. Requirements language . . . . .	4
2.2. I2RS Definitions . . . . .	4
3. Requirements control-plane datastore and ephemeral capabilities . . . . .	5
3.1. I2RS protocol requirements . . . . .	5
3.2. Overview of NETCONF support for I2RS protocol requirements . . . . .	5
4. NETCONF capability for control-plane datastore . . . . .	5
4.1. Overview . . . . .	6
4.2. Dependencies . . . . .	7
4.3. Capability identifier . . . . .	8
4.4. New Operations . . . . .	8
4.4.1. <get-data> . . . . .	8
4.4.2. <write data gt; . . . . .	10
4.5. Modification to protocol operations . . . . .	15
4.5.1. Unsupported protocol operations . . . . .	15
4.5.2. Modified protocol operations . . . . .	16
4.6. Interactions with Capabilities . . . . .	16
4.6.1. Unsupported Capabilities . . . . .	16
4.6.2. Modified Capabilities . . . . .	16
5. NETCONF Ephemeral capability . . . . .	17
5.1. Overview . . . . .	17
5.2. Dependencies . . . . .	18
5.3. New Operations . . . . .	18
5.3.1. resource-limits . . . . .	18
5.4. Modifications to Protocol Operations . . . . .	18
5.4.1. Unsupported Operations . . . . .	18
5.4.2. Modified Operations . . . . .	19
6. Yang model Simple Ephemeral Data model . . . . .	19
7. IANA Considerations . . . . .	22
8. Security Considerations . . . . .	22
9. Acknowledgements . . . . .	22
10. References . . . . .	22
10.1. Normative References: . . . . .	22
10.2. Informative References . . . . .	24
Authors' Addresses . . . . .	27

## 1. Introduction

This a proposal for Yang additions to support the first version of the I2RS protocol.

The I2RS architecture [RFC7921] defines the I2RS interface "a programmatic interface for state transfer in and out of the Internet routing system". The I2RS protocol is a protocol designed to a higher level protocol comprised of a set of existing protocols which have been extended to work together to support a new interface to the routing system. The I2RS protocol is a "reuse" management protocol which creates new management protocols by reusing existing protocols and extending these protocols for new uses, and has been designed to be implemented in phases [RFC7921].

The first version of the I2RS protocol is comprised of extensions to existing features of NETCONF [RFC6241] and RESTCONF [I-D.ietf-netconf-restconf]. The data modeling language for the I2RS protocol will be Yang [RFC7950] with features and extensions proposed in this draft.

The structure of this document is:

Section 2 provides definitions for terms in this document.

Section 3 summarizes the I2RS requirements behind these changes.

Section 4 describes the NETCONF capability to support a control protocol datastore.

Section 5 the NETCONF capability to support ephemeral state. [I-D.ietf-i2rs-ephemeral-state] specifies the I2RS requirements for the ephemeral state.

Section 6 provides a Tiny Routing Rib Yang module used by the examples in this document.

## 2. Definitions Related to Ephemeral Configuration

This section reviews definitions from I2RS architecture [RFC7921] and NETCONF operational state definitions [I-D.ietf-netmod-revised-datastores] before using these to construct a definition of the ephemeral data store.

## 2.1. 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].

## 2.2. I2RS Definitions

The I2RS architecture [RFC7921] defines the following terms:

**ephemeral data:** is data which does not persist across a reboot (software or hardware) or a power on/off condition. Ephemeral data can be configured data or data recorded from operations of the router. Ephemeral configuration data also has the property that a system cannot roll back to a previous ephemeral configuration state. (See [RFC7921] for an architectural overview, [I-D.ietf-i2rs-ephemeral-state] for requirements, and [I-D.ietf-netmod-revised-datastores] for discussion of how the ephemeral datastore as a control plane datastore interacts with intended datastore and dynamic configuration protocols to form the applied datastore".

**local configuration:** is the data on a routing system which does persist across a reboot (software or hardware) and a power on/off condition. Local configuration has the ability to roll back to a pervious configuration state. Local configuration is defined as the intended datastore [I-D.ietf-netmod-revised-datastores] which is modified by dynamic configuration protocols (such as DHCP) and the I2RS ephemeral data store

**dynamic configuration protocols datastore** are configuration protocols such as DHCP that interact with the intended datastore (which does persist across a reboot (software or hardware) power on/off condition), and the I2RS ephemeral state control plane datastore.

**operator-applied policy:** is a policy that an operator sets that determines how the ephemeral datastore as a control plane data store interacts with applied datastore (as defined in [I-D.ietf-netmod-revised-datastores]). This operator policy consists of setting a priority for each of the following (per [I-D.ietf-i2rs-ephemeral-state]):

- \* intended configuration,
- \* any dynamic configuration protocols,
- \* any control plane datastores (one of which is ephemeral.)

### 3. Requirements control-plane datastore and ephemeral capabilities

#### 3.1. I2RS protocol requirements

The requirements for the I2RS protocol are defined in the following documents:

- o I2RS Problem Statement [RFC7920],
- o I2RS Architecture [RFC7921],
- o I2RS Traceability [RFC7922],
- o Publication and Subscription [RFC7923],
- o I2RS Ephemeral State Requirements, ,  
[I-D.ietf-i2rs-ephemeral-state]
- o I2RS Protocol Security Requirements,  
[I-D.ietf-i2rs-protocol-security-requirements]

The Interface to the routing System (I2RS) creates a new capability for the routing systems, and with greater capabilities come a greater need for security. The requirements for a secure environment for I2RS are described in [I-D.ietf-i2rs-security-environment-reqs].

#### 3.2. Overview of NETCONF support for I2RS protocol requirements

This overview reviews the following:

- o Dependencies on Existing features
- o Additions to use NETCONF [RFC6241] to support control plane datastores changes get to get data, write data, (via target [merge, replace, create, delete, copy, delete-all]), close-session, kill-session, rollback-on-error (all-or-nothing), validate (validation + roll-back-on-error (all-or-nothing),
- o Additions for I2RS ephemeral
- o NETCONF [RFC6241] changes to obtain control-plane datastore.

### 4. NETCONF capability for control-plane datastore

capability-name: control-plane

#### 4.1. Overview

This capability defines the NETCONF protocol extensions for use with control plane protocols.

A control plane datastore is not part of the configuration datastore per [I-D.ietf-netmod-revised-datastores]. The control plane datastore may contain configuration and operational state. A router implementation may merge the configuration from a control plane datastore with configuration data from the configuration datastore. A query of the applied datastore will provide a list of the installed configuration from all datastores with meta data. The current architectural provides an origin identityref with the following mapping to datastores for the

- o static - configuration data store.
- o dynamic - dynamic configuration or dynamic control plane datastores.
- o system - created by system,
- o data-model - created by "default" in use.

Clearly, the dynamic origin title is not enough to uniquely identify a control plane datastore entry in the applied datastore. Additional definitions will need to be added to the architectural model, but this will be specified in another document.

The control plane datastores do not restrict multiple access via the locking mechanisms (<lock> and <unlock>), but use a priority scheme to handle multiple clients attempting to write the same data. The default validation within a control plane datastore's config objects (e.g. config=TRUE) is the configuration datastore validation, but if Yang data modules specify different validation for the datastore or specific nodes then the control plane datastores will use this validation.

Some data modules may be used for both a control plane datastore and the configuration datastore. If additional validation is used for these modules, it is recommended that these modules use the "rpc" function for the additional validation rather than the <write-data> functions.

#### 4.2. Dependencies

The following are the dependencies for the :control-plane capability

- o Yang features:

- \* [I-D.ietf-netmod-revised-datastores] functionality including the ietf-yang-architecture" data module.
- \* [I-D.hares-netmod-i2rs-yang] Yang additions related to datastores definitions related to control plane datastores (datastoredef, datastore, dstype, precedence, protosup, validation), and ephemeral state.

- o The following NETCONF features:

- \* NETCONF [RFC6241] with its updates [RFC7803],
- \* Network Access Control Model [RFC6536] with update by [I-D.ietf-netconf-rfc6536bis]
- \* Running NETCONF over TLS with mutually X.509 authentication [RFC7589]
- \* Keystore Model [I-D.ietf-netconf-keystore],
- \* Subscribing to Yang Datastore updates [I-D.ietf-netconf-yang-push],
- \* NETCONF support for Event Notifications [I-D.ietf-netconf-netconf-event-notifications],
- \* Subscribing to NETCONF Events (updated) [I-D.ietf-netconf-rfc5277bis]
- \* Yang Patch Media type [I-D.ietf-netconf-yang-patch],
- \* NETCONF/RESTCONF Zero Touch provisioning [I-D.ietf-netconf-zerotouch],
- \* TLS Client and Server Models [I-D.ietf-netconf-tls-client-server]
- \* Call Home [I-D.ietf-netconf-call-home],
- \* Module library [RFC7895],

- \* NETCONF/RESTCONF Zero Touch provisioning  
[I-D.ietf-netconf-zerotouch],

#### 4.3. Capability identifier

The controlplane-datastore capability is identified by the following capability string: (:control-plane (uri-tbd)) where the uri-tbd is to be assigned by IANA.

#### 4.4. New Operations

The following are additional protocol operations NETCONF [RFC6241] to support the following queries based on a datastore source/target datastore being specified:

- o "get-data"
- o "write-data"
- o "validate-data"

The <target-datastore> must be registered with IANA.

##### 4.4.1. <get-data>

The get-data command has obtains configuration and operational data. The parameters the following:

**source** name of the datastore being queried. The valid names are "applied", "opstate", or a datastore name registered with IANA.

**filter** this identifies the portions of the device configuration datastore is to receive. If this parameter is not present, the entire datastore is returned. The filter MAY support subtypes "subtree", "uri", and "xpath" capabilities described in [RFC6241]. Filters may also include the elements for state (E.g. config true, config false, ephemeral true; ephemeral false;).

**Positive Response** If the device was able to satisfy the request, an <rpc-reply> is sent. The <data> section contains the appropriate subset.

**Negative Response** If the device was unable to satisfy the request, an <rpc-error> is included in the <rpc-reply>



Example - retrieve route1 in route list.  
wfrom control plane datastore (cp-alpha)  
and gets both configuration and ephemeral data.

```
<rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-data/
    <source>
      <cp-alpha/>
    </source>
    <filter type="subtree">
      <top xmlns:t=
        "http://example.com/schema/1.0/i2rs/tiny-rt-instance">
        <route-list>
          <route-index>1</route-index>
        </route-list>
      </filter>
    </get-data>
  </rpc>

<rpc-rply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-data>
    <source>
      <cp-alpha/>
    </source>
    <filter type="subtree">
      <top xmlns:t=
        "http://example.com/schema/1.0/i2rs/tiny-rt-instance">
        <route-list>
          <route-index>1</route-index>
          <prefix-match>192.2/8 /16</prefix-match>
          <nexthop>129.1.5.1</nexthop>
          <if-outgoing>Eth0</if-outgoing>
          <installed>true</installed>
        </route-list>
      </filter>
    </get-data>
  </rpc>
```

Figure 1

Example 2 - retrieve users subtree from the ephemeral database which has example control plane datastore (cp-alpha) and gets only config=true data;

```
<rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-data/
    <source>
      <cp-alpha/>
    </source>
    <filter type="subtree">
      <top xmlns:t=
        "http://example.com/schema/1.0/i2rs/tiny-rt-instance">
        <route-list>
          <route-index>1</route-index>
        </route-list>
        type="state" select "config";
      </filter>
    </get-data>
  </rpc>

<rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-data>
    <source>
      <cp-alpha/>
    </source>
    <filter type="subtree">
      <top xmlns:t=
        "http://example.com/schema/1.0/i2rs/tiny-rt-instance">
        <route-list>
          <route-index>1</route-index>
          <prefix-match>192.2/8 /16</prefix-match>
          <nexthop>129.1.5.1</nexthop>
          <if-outgoing>Eth0</if-outgoing>
        </route-list>
      </filter>
    </get-data>
  </rpc>
```

Figure 2 - get config data

#### 4.4.2. <write data gt;

The write data operational has the attributes of operation parameters, and parameters of target datastore, default-operations, test-option, error-option, a priority, secondary-id, config, and

opstate. The attributes of the operation for individual nodes wutg "config-true" are: create, delete, merge, remove, and replace. The attributes for all-datastore operations are create-datastore, copy-datastore, delete-datastore. The operations handle multiple-client writes by using priority values rather than a locking mechanism. If two or more clients interact over changing the data node, the priority values arbitrate between the the clients where the greatest priority (1=lowest) wins. The following operations can enact changes in opstate data nodes: create, delete, remove, reset.

The default-operations parameter flags are: merge, replace, or none. The test-option parameters flags are: test-then-set, set, and test-only. The error-option oarameter flags are: stop-on-error, continue-on-error, and rollback-on-error. The priority parameter is a integer giving the priority (range 1..5000).

#### 4.4.2.1. Limitations based on other capability flags

The test-option parameters MAY only be set if the device advertises the :validate:1.1 capability. If test-option is set without the :validate:1.1 capability, an error is returned "no support for test-option".

The error-option subparameter "rollback-on-error" is enabled only if the :rollback-on-error capability is set and the data is under the config parameter.

A URL is accepted within the <source> or <target> parameters if the :url capability is set. An XPATH is accepted within the <source> or <target> parameters if the :xpath capability is set.

#### 4.4.2.2. defaults

The following are the defaults:

- o The target datastore does not exists, it will be created if local support exists. Otherwise, the reply will indicate "non-supported datastore".
- o If no sub-operations is specified the sub-operation, the default is merge.
- o If no priority parameter is specified, the priority will be set to 1 (lowest external priority).
- o If error-option is not set, then the default is "stop-on-error". (Note: for I2RS WG input. Is "stop-on-error" the same as "none"? )

- o If no test-option parameter is set, the write data operates as a 'set' without validation first.
- o if no secondary-identifier is given, the secondary identifier stored is "" indicating a null string.

The NETCONF priority for the "write data" function is simply the Netconf priority range. If implementations have an internal priority, the precedence between the local configuration and the NETCONF supplied configuration is set by a operator applied knob. For example, an implementation could indicate that the local configuration priority can range from 0-10 and the NETCONF priority is 10 plus the value of the priority parameter.

#### 4.4.2.3. target

The target is a datastore whose name is registered with IANA.

Example Write data

dsname = i2rs-agent

```
<rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <write data>
    <target><i2rs-agent></target>
    <operation>merge</operation>
    <priority>50</priority>
    <error-option>all-or-nothing</error-option>
    <config>
      <top xmlns:t=
        "http://example.com/schema/1.0/i2rs/tiny-rt-instance">
        <route-list>
          <route-index>1</route-index>
          <prefix-match>192.2/8 /16<prefix-match>
          <nexthop>129.1.5.1</nexthop>
          <if-outgoing>Eth0</if-outgoing>
        </route-list>
      </top>
    </config>
  </write data>
```

#### 4.4.2.4. write data operations attributes

The description of each operation is below:

`<create>` (config, opstate) : the creation of the data node in the datastore if and only if the data does not already exist in the target datastore. If the data already exists, then an `<rpc-error>` is return wtih an error tag of "data-exists". Failure information or Success informatnoi is also pass to the notification functions (events and traceability).

`<create-datastore >` (config, opstate) : the creation of the datastore based on datastructures in the config and opstate parameters. The datastore ownership is set to client creating the datastore plus the priority.

`<delete>` (config) : the deletion of the data node if the data node exists in the configuration and either the same client deletes the node or a client with a high priority deletes the node. If configuration data does not exist in the datastore, then the `<rpc-error>` element is returned with a `<error-tag>` with value of "data-missing". The error information is passed to the notification functions to be sent as event or (optionally) placed in a tracing file.

`<delete-datastore>` Delete all configuration and operational data configured into the datastore, and the delete the datastore. The client requesting a delete-store must either be the owner of the datastore or have a higher priority than the client that owns the datastore. If a higher priority client takes ownership, the lower priority client is notified. If the devices is able to satisfy request, the positive response is `<rcp-reply>` that includes `<ok>` element. If the device is unable to complete request, the `<rcp-reply>` that includes `<rpc-error>` element. The operations results are forwarded to event and traceabilty functions.

`<copy-datastore>` If the datastore does not exist, it creates the datastore and copies the configuration values and opstate values into the datastore. The ownership information (client identity and priority) is saved as part of the datastore. If the datastore does exist and the client with ownership of the datastore changes it, then the client can replace all the datastore nodes. If a different client with lower priority than the client having ownership wants to change the datastore, the request is rejected. If a client with higher priority than the client having ownership, then the the owership changed to the new client, all the data in the datastore is deleted, all new data uploaded (config and opstate nodes). If a server is able to satisfy request, the

positive response is <rcp-reply> that includes <ok> element. If the server is unable to complete request, the <rcp-reply> that includes <rpc-error> element. The operational results are forwarded to event and traceability functions. If a copy-datastore action is in progress, and a client with a higher priority asks to copy-datastore, the original

<merge> (config) : parameter specifies merge. If the <priority> specifies The current data is modified by the new data in a merge of the data based on priorities. If the same client merges the data, priority is ignored. If a different client merges the data, the priority must be created than the current client's priority. If any data is replaced, this event is passed to the notification and traceability functions to pass to the setting client and the client that set the original value.

<remove> (config, opstate) : the remove of the data node if the data nodes specified in the <config> or the <opstate> node exist. If data nodes do not exist, the "remove" operation is silently ignored and error results are forwarded to traceability functions.

<replace> (config) : replaces data in target if the same client replaces the top-level node, priority is ignored. If a different client replaces the note, the priority must be higher than the top level node's priority. If any data is replaced, this event is passed to the notification function (events and traceability), and a notification is sent to the previous client setting this data that the data has been reset. If the request to replace is reject due to the current top-level node having a higher priority, then an <rpc-error> returns with an error tag of "insufficient-priority". If the node is replace by a different client, the original client is notified of the change.

<reset> (opstate) : resets opstate nodes with counters to initial settings.

#### 4.4.2.5. <priority parameters>

The priority parameter sets a integer value for the priority as shown in figure x.

#### 4.4.2.6. <test-op parameter>

The <test-option> parameter performs basic function it does in the <edit-config> basic function. Just in [RFC6241], the <test-option> parameter MAY be specified only if the device advertises the ":validate:1.1" capability. The only difference is that the validation specified by the data model may augment the validation

test and the validation will also include the ability of the client to set this element. If a validation error occurs, the test-then-set will not perform the write-data function.

#### 4.4.2.7. <error-option> parameter

<error-option> has the following attributes

stop-on-error : Abort the write-data on the first error. This is the default.

msg-rollback-on-error : if an error condition occurs such that error severity <rpc-error> is generated, the server will stop processing the write-data operation and restore the specific configuratoin to its complete state at the start of the "write-data" operation. This option only processes roll-back on single messages which includes

- \* if multiple operations occur in single message, error in one operation (E.g. read data) must not impact other operations (write-data);
- \* multiple operations in multiple message should be supported, but roll-back should only include a single message.

This option requires the server to support the :rollback-on-error capability.

#### 4.4.2.8. secondary-id

This operation associates a secondary identifier with a set of write-data operations. The secondary identifier is an opaque string.

### 4.5. Modification to protocol operations

#### 4.5.1. Unsupported protocol operations

The following protocol operations are not supported in the control plane datastore:

- o <get-config>,
- o <edit-config>,
- o <copy-config>,
- o <delete-config>,

- o <lock>,
- o <unlock>

#### 4.5.2. Modified protocol operations

##### 4.5.2.1. <close-session> and <kill-session>

The <close-session> is modified to take a target of a control plane datastore name (registered with IANA). Since no locks are set, none should be released.

The <kill-session> is modified to take a target of a control plane datastore name (registered with IANA). Since no locks are set, none should be released.

#### 4.6. Interactions with Capabilities

##### 4.6.1. Unsupported Capabilities

The following capabilities are not supported:

- writeable-running capability,
- candidate configuration capability,
- confirmed commit capability,
- distinct startup capability,

##### 4.6.2. Modified Capabilities

###### 4.6.2.1. rollback-on-error

The rollback-on-error allows the error handling to be roll-back-on-error (all-or-nothing in I2RS terms) for the control plane datastore. The control plane datastore name is a valid target if the rollback-on-error capability is combined with the control plane datastore capability.

###### 4.6.2.2. validate

The validation capability engaged with the control plane capability operates to validate the config portion of the control plane datastore. Therefore, the <target> is allowed to have a datastore name which is registered with IANA.



The validation of the configuration portion may contain the "validation" yang command which provides alternative validation mechanisms for specific data objects.

#### 4.6.2.3. URL capability and XPATH capability

The URL capabilities specify a <url> in the <source> and <target> operate as normal, but are allowed to specify a module within a control plane datastore.

### 5. NETCONF Ephemeral capability

capability-name: control-plane

#### 5.1. Overview

The ephemeral capability is the ability to support control plane datastores which are entirely ephemeral or have ephemeral state modules, or ephemeral statements within objects in a modules. These objects can be configuration state (config=TRUE) or operational state (config=FALSE).

Ephemeral state in datastores, ephemeral modules or ephemeral objects within a module have one key characteristics: the data does not persist across reboots. The ephemeral configuration state must be restored by a client, and the operational state will need to be regenerated.

The entire requirements for ephemeral state for the I2RS control plane protocol are listed in [I-D.ietf-i2rs-ephemeral-state]. Many of these require fulfilled by the NETCONF control-plane capability(Ephemeral-REQ-07, Ephemeral-REQ-11, Ephemeral-REQ-12, Ephemeral-REQ-13, Ephemeral-REQ-14, Ephemeral-REQ-16).

The key features include:

- o references between (to/from) ephemeral state and non-ephemeral state for constraints purposes (see Ephemeral-REQ-02, Ephemeral-REQ-03, and Ephemeral-REQ-04 in [I-D.ietf-i2rs-ephemeral-state]).
- o operations to set and modify the constraints on the amount of resources the I2RS Agent (aka NETCONF server) can consume (Ephemeral-REQ-05)
- o Yang modules must identify Yang objects (modules, submodules or objects within yang modules which are ephemeral and augment other nodes) and allow an "ephemeral=TRUE" feature.

## 5.2. Dependencies

Ephemeral state is not supported in the configuration datastore. The ephemeral state capability depends on having the control-plane datastore capability enabled (with appropriate NETCONF capabilities described above), and an IANA registered datastore name.

Yang must support the ability to denote that a datastore, module, submodule or object within a module can be denoted as ephemeral. This capability depends on the yang additions described in [I-D.hares-netmod-i2rs-yang] for control plane datastores, ephemeral key word, and validation key word.

Ephemeral state operation depends on notification of events and traceability of errors. I2RS ephemeral state requires that

## 5.3. New Operations

Note: One operation that is suggested for ephemeral state is to set resource limits. It does not seem to be an ephemeral state issue, but a control plane issue. This feature is placed here until future discussion for I2RS WG.

### 5.3.1. resource-limits

resource-limits

definition - TBD

The [I-D.ietf-i2rs-ephemeral-state] suggests setting these limits, but it does not seem to be an ephemeral function.

## 5.4. Modifications to Protocol Operations

### 5.4.1. Unsupported Operations

The ephemeral state only works as an augment to the control-plane datastore. Therefore, the following protocol operations, which are not supported in the control-plane datastore capability, are also not supported in the ephemeral capability:

- o <get-config>,
- o <edit-config>,
- o <copy-config>,
- o <delete-config>,

- o <lock>,
- o <unlock>

#### 5.4.2. Modified Operations

The ephemeral state only works as an augment to the control-plane datastore with specific ephemeral validations. Therefore, the <close-session> and <kill-session> are modified as described in the sections below.

##### 5.4.2.1. <close-session> Modifications

The <close-session> is modified to take a target of a control plane datastore name (registered with IANA). Since no locks are set, none should be released.

##### 5.4.2.2. <kill-session> Modifications

The <kill-session> is modified to take a target of a control plane datastore name (registered with IANA). Since no locks are set, none should be released.

##### 5.4.2.3. <validate> Modifications

The ephemeral state may require validation to determine if the constraints obey ephemeral-state rules. If the :validate capability is used, the following parameter requires ephemeral-state constraints (Ephemeral-REQ-02, Ephemeral-REQ-03, and Ephemeral-REQ-04). If the ephemeral-constraint parameter is engaged for a module or object that is not ephemeral, the parameter is silently ignored. Error information is forwarded to the event notification processes and the traceability functions.

Additional Parameter

ephemeral-constraint

#### 6. Yang model Simple Ephemeral Data model

```
datastoredef cp-alpha {
  dtype control-plane;
  description {
    "example control plane datastore ";
  }
  module-list tiny-rt-instance;
  precedence applied {
    precedenceval 5;
  }
}
```

```
    }
    precedence opstate {
        precedenceval 5;
    }
}

datastoredef cp-beta {
    dstype control-plane i2rs-v0;
    description {
        "example control plane datastore ";
    }
    module-list tiny-rib;
    ephemeral true;
    precedence applied {
        precedenceval 50;
    }
    precedence opstate {
        precedenceval 50;
    }
}

module cp-example-1 {
    namespace "http://exaple.com/schema/cp-examples/1.1/tiny-rib";
    prefix trib;
    import ietf-inet-types {
        prefix inet;
    }
    import ietf-yang-types {
        prefix yang;
    }

    grouping trib-rt {
        description "tiny rib route";
        leaf route-index {
            type uint64;
            mandatory true;
            description "route index";
        }
        leaf v4-prefix-match {
            type inet:ipv4-prefix;
            mandatory true;
        }
        leaf v4-nexthop {
            type inet:ipv4;
            mandatory true;
        }
        leaf if-outgoing {
            type if:interface-ref;
            mandatory true;
        }
    }
}
```

```
        description {
            "Name of outgoing interface";
        }
    leaf installed {
        type boolean;
    config false;
        description "rt install status ";
    }
}
container tiny-rt-instance {
    description
        "Tiny routing instance for
        example purposes";
    leaf name {
        type string;
        description
            "The name of routing instance
            which must be unique. ";
    }
    list route-list {
        key "route-index";
        description
            "a list of routes of rib"
        uses trib-rt;
    }
}

rpc trib-add {
    description "add route to tiny rib";
    input {
        leaf datastore {
            type string; //iana registered
            mandatory true;
            description
                "iana datastore name";
        }
        container trib-routes {
            description
                "Tiny rib routes to be added
                to tiny rib";
            list route-list {
                key "route-index";
                users trib-rt;
            }
        }
    }
    output {
        container trib-add-status {
```

```
        leaf success {  
            type boolean;  
            description "add succeeded";  
        }  
        leaf failure-reason {  
            type string;  
            description "reason for failure ";  
        }  
    }  
}
```

## 7. IANA Considerations

TBD

## 8. Security Considerations

The security requirements for the I2RS protocol are covered in [I-D.ietf-i2rs-protocol-security-requirements]. The security environment the I2RS protocol is covered in [I-D.ietf-i2rs-security-environment-reqs]. Any person implementing or deploying the I2RS protocol should consider both security requirements.

## 9. Acknowledgements

TBD

## 10. References

### 10.1. Normative References:

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC4107] Bellovin, S. and R. Housley, "Guidelines for Cryptographic Key Management", BCP 107, RFC 4107, DOI 10.17487/RFC4107, June 2005, <<http://www.rfc-editor.org/info/rfc4107>>.
- [RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <<http://www.rfc-editor.org/info/rfc4960>>.

- [RFC5339] Le Roux, J.L., Ed. and D. Papadimitriou, Ed., "Evaluation of Existing GMPLS Protocols against Multi-Layer and Multi-Region Networks (MLN/MRN)", RFC 5339, DOI 10.17487/RFC5339, September 2008, <<http://www.rfc-editor.org/info/rfc5339>>.
- [RFC5424] Gerhards, R., "The Syslog Protocol", RFC 5424, DOI 10.17487/RFC5424, March 2009, <<http://www.rfc-editor.org/info/rfc5424>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<http://www.rfc-editor.org/info/rfc6020>>.
- [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, <<http://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, <<http://www.rfc-editor.org/info/rfc6242>>.
- [RFC6244] Shafer, P., "An Architecture for Network Management Using NETCONF and YANG", RFC 6244, DOI 10.17487/RFC6244, June 2011, <<http://www.rfc-editor.org/info/rfc6244>>.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <<http://www.rfc-editor.org/info/rfc6536>>.
- [RFC7158] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7158, DOI 10.17487/RFC7158, March 2014, <<http://www.rfc-editor.org/info/rfc7158>>.
- [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, <<http://www.rfc-editor.org/info/rfc7589>>.
- [RFC7803] Leiba, B., "Changing the Registration Policy for the NETCONF Capability URNs Registry", BCP 203, RFC 7803, DOI 10.17487/RFC7803, February 2016, <<http://www.rfc-editor.org/info/rfc7803>>.

- [RFC7895] Bierman, A., Bjorklund, M., and K. Watsen, "YANG Module Library", RFC 7895, DOI 10.17487/RFC7895, June 2016, <<http://www.rfc-editor.org/info/rfc7895>>.
- [RFC7920] Atlas, A., Ed., Nadeau, T., Ed., and D. Ward, "Problem Statement for the Interface to the Routing System", RFC 7920, DOI 10.17487/RFC7920, June 2016, <<http://www.rfc-editor.org/info/rfc7920>>.
- [RFC7921] Atlas, A., Halpern, J., Hares, S., Ward, D., and T. Nadeau, "An Architecture for the Interface to the Routing System", RFC 7921, DOI 10.17487/RFC7921, June 2016, <<http://www.rfc-editor.org/info/rfc7921>>.
- [RFC7922] Clarke, J., Salgueiro, G., and C. Pignataro, "Interface to the Routing System (I2RS) Traceability: Framework and Information Model", RFC 7922, DOI 10.17487/RFC7922, June 2016, <<http://www.rfc-editor.org/info/rfc7922>>.
- [RFC7923] Voit, E., Clemm, A., and A. Gonzalez Prieto, "Requirements for Subscription to YANG Datastores", RFC 7923, DOI 10.17487/RFC7923, June 2016, <<http://www.rfc-editor.org/info/rfc7923>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<http://www.rfc-editor.org/info/rfc7950>>.
- [RFC7952] Lhotka, L., "Defining and Using Metadata with YANG", RFC 7952, DOI 10.17487/RFC7952, August 2016, <<http://www.rfc-editor.org/info/rfc7952>>.
- [RFC7958] Abley, J., Schlyter, J., Bailey, G., and P. Hoffman, "DNSSEC Trust Anchor Publication for the Root Zone", RFC 7958, DOI 10.17487/RFC7958, August 2016, <<http://www.rfc-editor.org/info/rfc7958>>.

## 10.2. Informative References

- [I-D.hares-netmod-i2rs-yang]  
Hares, S. and a. amit.dass@ericsson.com, "Yang for I2RS Protocol", draft-hares-netmod-i2rs-yang-04 (work in progress), March 2017.
- [I-D.ietf-i2rs-ephemeral-state]  
Haas, J. and S. Hares, "I2RS Ephemeral State Requirements", draft-ietf-i2rs-ephemeral-state-23 (work in progress), November 2016.



- [I-D.ietf-i2rs-protocol-security-requirements]  
Hares, S., Migault, D., and J. Halpern, "I2RS Security Related Requirements", draft-ietf-i2rs-protocol-security-requirements-17 (work in progress), September 2016.
- [I-D.ietf-i2rs-rib-data-model]  
Wang, L., Ananthakrishnan, H., Chen, M., amit.dass@ericsson.com, a., Kini, S., and N. Bahadur, "A YANG Data Model for Routing Information Base (RIB)", draft-ietf-i2rs-rib-data-model-07 (work in progress), January 2017.
- [I-D.ietf-i2rs-rib-info-model]  
Bahadur, N., Kini, S., and J. Medved, "Routing Information Base Info Model", draft-ietf-i2rs-rib-info-model-10 (work in progress), December 2016.
- [I-D.ietf-i2rs-security-environment-reqs]  
Migault, D., Halpern, J., and S. Hares, "I2RS Environment Security Requirements", draft-ietf-i2rs-security-environment-reqs-03 (work in progress), March 2017.
- [I-D.ietf-i2rs-yang-l3-topology]  
Clemm, A., Medved, J., Varga, R., Liu, X., Ananthakrishnan, H., and N. Bahadur, "A YANG Data Model for Layer 3 Topologies", draft-ietf-i2rs-yang-l3-topology-08 (work in progress), January 2017.
- [I-D.ietf-netconf-call-home]  
Watsen, K., "NETCONF Call Home and RESTCONF Call Home", draft-ietf-netconf-call-home-17 (work in progress), December 2015.
- [I-D.ietf-netconf-keystore]  
Watsen, K. and G. Wu, "Keystore Model", draft-ietf-netconf-keystore-00 (work in progress), October 2016.
- [I-D.ietf-netconf-netconf-event-notifications]  
Prieto, A., Clemm, A., Voit, E., Nilsen-Nygaard, E., Tripathy, A., Chisholm, S., and H. Trevino, "NETCONF Support for Event Notifications", draft-ietf-netconf-netconf-event-notifications-01 (work in progress), October 2016.
- [I-D.ietf-netconf-restconf]  
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf-18 (work in progress), October 2016.

- [I-D.ietf-netconf-rfc5277bis]  
Clemm, A., Prieto, A., Voit, E., Nilsen-Nygaard, E., Tripathy, A., Chisholm, S., and H. Trevino, "Subscribing to Event Notifications", draft-ietf-netconf-rfc5277bis-01 (work in progress), October 2016.
- [I-D.ietf-netconf-rfc6536bis]  
Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", draft-ietf-netconf-rfc6536bis-00 (work in progress), January 2017.
- [I-D.ietf-netconf-tls-client-server]  
Watsen, K., "TLS Client and Server Models", draft-ietf-netconf-tls-client-server-01 (work in progress), November 2016.
- [I-D.ietf-netconf-yang-patch]  
Bierman, A., Bjorklund, M., and K. Watsen, "YANG Patch Media Type", draft-ietf-netconf-yang-patch-14 (work in progress), November 2016.
- [I-D.ietf-netconf-yang-push]  
Clemm, A., Voit, E., Prieto, A., Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "Subscribing to YANG datastore push updates", draft-ietf-netconf-yang-push-05 (work in progress), March 2017.
- [I-D.ietf-netconf-zerotouch]  
Watsen, K. and M. Abrahamsson, "Zero Touch Provisioning for NETCONF or RESTCONF based Management", draft-ietf-netconf-zerotouch-12 (work in progress), January 2017.
- [I-D.ietf-netmod-revised-datastores]  
Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "A Revised Conceptual Model for YANG Datastores", draft-ietf-netmod-revised-datastores-00 (work in progress), December 2016.
- [I-D.ietf-netmod-schema-mount]  
Bjorklund, M. and L. Lhotka, "YANG Schema Mount", draft-ietf-netmod-schema-mount-04 (work in progress), March 2017.
- [I-D.ietf-netmod-syslog-model]  
Wildes, C. and K. Koushik, "A YANG Data Model for Syslog Configuration", draft-ietf-netmod-syslog-model-12 (work in progress), February 2017.

Authors' Addresses

Susan Hares  
Huawei  
Saline  
US

Email: shares@ndzh.com

Amit Daas  
Ericsson

Email: amit.dass@ericsson.com

I2RS working group  
Internet-Draft  
Intended status: Informational  
Expires: September 14, 2017

S. Hares  
Huawei  
A. Dass  
Ericsson  
March 13, 2017

RESTCONF Changes to Support I2RS Protocol  
draft-hares-netconf-i2rs-restconf-01.txt

Abstract

This document describes two RESTCONF optional capabilities (control plane datastore capability, ephemeral state capabilities) that are needed to support the I2RS protocol needs. The I2RS protocol requires an ephemeral control plane datastore. as control plane datastores.

The purpose of this draft is to kick-start the discussions with NETCONF on these two capabilities.

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 <http://datatracker.ietf.org/drafts/current/>.

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

This Internet-Draft will expire on September 14, 2017.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
1.1. Background on I2RS . . . . .	3
1.2. Structure of draft . . . . .	3
2. Definitions and Background on I2RS . . . . .	3
2.1. IETF Requirements language . . . . .	3
2.2. I2RS Definitions . . . . .	3
2.3. Example for operator-applied policy . . . . .	5
2.4. I2RS protocol requirements . . . . .	6
3. RESTCONF control plane datastore capability . . . . .	6
3.1. Overview . . . . .	6
3.2. Dependencies . . . . .	7
3.3. New Operations . . . . .	7
3.4. Modified Operations . . . . .	7
4. RESTCONF protocol extensions for the ephemeral datastore . .	8
4.1. Overview . . . . .	8
4.2. Dependencies . . . . .	9
4.3. Capability identifier . . . . .	9
4.4. New Operations . . . . .	9
4.5. modification to data resources . . . . .	9
4.6. Modification to existing operations . . . . .	10
5. IANA Considerations . . . . .	10
6. Security Considerations . . . . .	10
7. Acknowledgements . . . . .	11
8. References . . . . .	11
8.1. Normative References: . . . . .	11
8.2. Informative References . . . . .	13
Authors' Addresses . . . . .	16

## 1. Introduction

This a proposal for the following two RESTCONF capabilities to augment RESTCONF [RFC8040] to support the first version of the I2RS protocol: Control plane datastore capability and ephemeral state capability. The yang that supports this proposal is described in [I-D.hares-netmod-i2rs-yang]. This work is based on the datastore definitions in [I-D.ietf-netmod-revised-datastores].

This draft parallels a similar proposal for NETCONF [RFC6241] is described in [I-D.hares-netconf-i2rs-protocol]. The difference is the original design is to embedded the I2RS multi-headed collision resolution in the "control plane data store capability". However

RESTCONF has edit-collision capability already which only needs a usage description. Therefore, this document has a I2RS Edit-Collision capability.

Caveat: This work is an individual draft (not an I2RS WG effort)

### 1.1. Background on I2RS

The I2RS architecture [RFC7921] defines the I2RS interface "a programmatic interface for state transfer in and out of the Internet routing system". The I2RS protocol is a protocol designed to a higher level protocol comprised of a set of existing protocols which have been extended to work together to support a new interface to the routing system. The I2RS protocol is a "reuse" management protocol which creates new management protocols by reusing existing protocols and extending these protocols for new uses, and has been designed to be implemented in phases [RFC7921].

### 1.2. Structure of draft

The structure of this document is:

Section 2 provides definitions and background on I2RS work. (If you are familiar with the I2RS architecture and requirements, you can skip this section.)

Section 3 describes the RESTCONF control plane datastore capability.

Section 4 describes the RESTCONF ephemeral state capability. .

## 2. Definitions and Background on I2RS

This section reviews definitions from I2RS architecture, and provides background on I2RS work for the reader.

### 2.1. IETF 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].

### 2.2. I2RS Definitions

The I2RS architecture [RFC7921] defines the following terms:

ephemeral data: is data which does not persist across a reboot (software or hardware) or a power on/off condition. Ephemeral

data can be configured data or data recorded from operations of the router. Ephemeral configuration data also has the property that a system cannot roll back to a previous ephemeral configuration state. (See [RFC7921] for an architectural overview, [I-D.ietf-i2rs-ephemeral-state] for requirements, and [I-D.ietf-netmod-revised-datastores] for discussion of how the ephemeral datastore as a control plane datastore interacts with intended datastore and dynamic configuration protocols to form the applied datastore".

local configuration: is the data on a routing system which does persist across a reboot (software or hardware) and a power on/off condition. Local configuration has the ability to roll back to a pervious configuration state. Local configuration is defined as the intended datastore [I-D.ietf-netmod-revised-datastores] which is modified by dynamic configuration protocols (such as DHCP) and the I2RS ephemeral data store

dynamic configuration protocols datastore are configuration protocols such as DHCP that interact with the intended datastore (which does persist across a reboot (software or hardware) power on/off condition), and the I2RS ephemeral state control plane datastore.

control plane protocols datastore is a datastore which is loaded by control plane protocols (e.g. I2RS protocol) rather than system configuration protocols. (see [I-D.ietf-netmod-revised-datastores]).

operator-applied policy: is a policy that an operator sets that determines how the ephemeral datastore as a control plane data store interacts with applied datastore (as defined in [I-D.ietf-netmod-revised-datastores]). This operator policy consists of policy knobs that the operator sets to determine how the I2RS agent control plane ephemeral state datastore will interact with the intended configuration datastor and the dynamic configuration protocol datastore. Three policy knobs could be used to implement this policy:

- \* policy knob 1: I2RS Ephemeral control-plane datastore takes takes precedence over the intended datastore in the routing protocols.
- \* policy knob 2: Updated intended configuration datastore takes precedence over the I2RS ephemeral control-plane data store in the routing protocols

- \* policy knob 3: Ephemeral control plane datastore takes precedence over any other dynamic configuration protocols datastore.

### 2.3. Example for operator-applied policy

An practical example for three states of the operator-applied policy may help the reader understand the concept. Consider the following three desired outcomes with their policy knob states:

Monitoring Features only    The policy knob settings are:

```
Policy knob 1=false,  
  
policy knob 2=true,  
  
Policy knob 3=false,
```

Action: I2RS protocol software feature is installed, but the operator does not want the I2RS ephemeral datastore to take precedence (that is be used) on any variables in the applied configuration datastore. This policy set might be valid if I2RS is only suppose to monitor data on this node through newly defined parameters.

I2RS Agent Changes win    the policy knob settings would be:

```
Policy knob 1=true,  
  
policy knob 2=false,  
  
Policy knob 3=false,
```

Action: This is the normal case for the I2RS Agent where the ephemeral control-plane datastore takes precedence over the intended configuration datastore and dynamic configuration datastores. The values from the I2RS ephemeral datastore are used rather than the intended configuration datastore and the dynamic configuration protocol datastore. When the ephemeral data is removed by the I2RS agent, the dyanmic configuration datastore and the intended configuration datastore state is restored, combined and passed to the routing protocols for application.

Just change until next configuration update    the policy knob settings would be:

```
Policy knob 1=true,
```



policy knob 2=true,

Policy knob 3=false,

Action: This case can occur if the I2RS Client write to the ephemeral control plane data store is only suppose to take precedence until the next configuration cycle from a centralized system. Suppose the local configuration is get by the centralized system at 11:00pm each night. The I2RS Client writes temporary changes to the routing system via the I2RS agent ephemeral write. At 11:00pm, the local configuration update overwrite the ephemeral. The I2RS Agent notifies the I2RS Client which is tracking which of the ephemeral changes are being overwritten.

## 2.4. I2RS protocol requirements

The requirements for the I2RS protocol are defined in the following documents:

- o I2RS Problem Statement [RFC7920],
- o I2RS Architecture [RFC7921],
- o I2RS Traceability [RFC7922],
- o Publication and Subscription [RFC7923],
- o I2RS Ephemeral State Requirements, ,  
[I-D.ietf-i2rs-ephemeral-state]
- o I2RS Protocol Security Requirements,  
[I-D.ietf-i2rs-protocol-security-requirements]

The Interface to the routing System (I2RS) creates a new capability for the routing systems, and with greater capabilities come a greater need for security. The requirements for a secure environment for I2RS is described in [I-D.ietf-i2rs-security-environment-reqs].

## 3. RESTCONF control plane datastore capability

capability-name: control-plane datastore

### 3.1. Overview

The :control-plane datastore capability enables the RESTCONF to support the following:

- o API resource that is {+restconf/cp-data} - the storage of control plane datastore's configuration that includes configuration ({+restconf/cp-data/config}) and operational state specific to the control plane datastore ({+restconf/cp-data/opstate}).
- o It also includes the ability to have the applied datastore and the opstate datastore (per [I-D.ietf-netmod-revised-datastores]).

### 3.2. Dependencies

This protocol strawman utilizes the following existing proposed features for NETCONF and RESTCONF

- o RESTCONF [RFC8040].
- o Module library [RFC7895],
- o RESTCONF Patch Media Type [RFC8072],
- o NETCONF Support for event notifications [I-D.ietf-netconf-netconf-event-notifications],
- o Publication/Subscription via Push [I-D.ietf-netconf-yang-push],
- o NETCONF and HTTP Transport for Event Notifications [I-D.ietf-netconf-restconf-notif],
- o Publication/Subscription via Push [I-D.ietf-netconf-yang-push],
- o syslog yang module (both [RFC5424] and [I-D.ietf-netmod-syslog-model])

### 3.3. New Operations

none

### 3.4. Modified Operations

All RESTCONF methods (OPTIONS, HEAD, GET, POST, PUT, PATCH, DELETE) need to work in the control plane datastores. config=TRUE data, and where appropriate config=FALSE data.

Editor's Note: Amazingly, RESTCONF is almost ready for the control plane data. The authors understanding of the I2RS protocol needs is why. The best situation is to ask the RESTCONF authors for help specifying what needs to change for the RESTCONF to allow references to datastore.

#### 4. RESTCONF protocol extensions for the ephemeral datastore

capability-name: ephemeral-state

##### 4.1. Overview

This capability defines the RESTCONF protocol extensions for control plane protocols that support control plane data stores with ephemeral data.

Ephemeral state is not unique to I2RS work.

The ephemeral capability is the ability to support control plane datastores which are entirely ephemeral or have ephemeral state modules, or ephemeral statements within objects in a modules. These objects can be configuration state (config=TRUE) or operational state (config=FALSE).

Ephemeral state in datastores, ephemeral modules or ephemeral objects within a module have one key characteristics: the data does not persist across reboots. The ephemeral configuration state must be restored by a client, and the operational state will need to be regenerated.

The entire requirements for ephemeral state for the I2RS control plane protocol are listed in [I-D.ietf-i2rs-ephemeral-state]. Compared to RESTCONF functionality there are 4 groups of additional changes:

**Constraints** The ability to enforce the constraints for references (to/from) the {+restconf/data} datastore, and a {+restconf/cp-data} control plane datastore. ((see Ephemeral-REQ-02, Ephemeral-REQ-03, and Ephemeral-REQ-04 in [I-D.ietf-i2rs-ephemeral-state])). The "validation" yang statement in [I-D.hares-netmod-i2rs-yang] could encode specific validation for the ephemeral case per datastore or per object. [Editor's note: Aid is needed from NETCONF authors to determine the best way to enforce the constraints.]

**Library Tracking of Ephemeral** Yang modules must identify Yang objects (modules, submodules or objects within yang modules which are ephemeral and augment other nodes) and allow an "ephemeral=TRUE" feature.

**Roll-back** an ephemeral node cannot roll-back to its previous value,

#### 4.2. Dependencies

The ephemeral capabilities have the following dependencies:

- o Yang modules must support the following:
  - \* identifying datastores, modules, and objects as ephemeral. (ephemeral=True)
  - \* Ability to have control plane datastores which are ephemeral.
- o The following features must be supported by RESTCONF
  - \* Module library [RFC7895],
  - \* RESTCONF Protocol [RFC8040],
  - \* RESTCONF Patch Media Type [RFC8072],
  - \* NETCONF Support for event notifications [I-D.ietf-netconf-netconf-event-notifications],
  - \* Publication/Subscription via Push [I-D.ietf-netconf-yang-push],
  - \* NETCONF and HTTP Transport for Event Notifications [I-D.ietf-netconf-restconf-notif],
  - \* Subscribing to Yang datastore push updates [I-D.ietf-netconf-yang-push],

#### 4.3. Capability identifier

The ephemeral-datastore capability is identified by the following capability string: ephemeral (TBD URI)

#### 4.4. New Operations

none

#### 4.5. modification to data resources

RESTCONF must be able to support the ephemeral data in a control plane datastore

RESTCONF library functions must be able to store an indication that a data module has ephemeral state.

#### 4.6. Modification to existing operations

The current operations in RESTCONF are: OPTIONS, HEAD, GET, POST, PUT, PATCH, and DELETE.

Editor's note: From here is not solutions but a list of features to discuss with the RESTCONF team.

The operations must support the following things about ephemeral The ephemeral data store has the following general qualities:

1. The ephemeral datastore is never locked. (RESTCONF does not use a locking mechanism.)
2. The ephemeral portion of the intended configuration, applied state, and derived state does not persist over a reboot,
3. an ephemeral node cannot roll-back to its previous value,
4. Since ephemeral data store is just data that does not persist over a reboot, then in theory any node or group of nodes in a YANG data model could be ephemeral. The YANG data module must indicate what portion of the data model (if any) is ephemeral.
  - \* A YANG data module could be all ephemeral (e.g. [I-D.ietf-i2rs-rib-data-model]) with no directly associated configuration models,
  - \* A YANG model could be all ephemeral but associated with a configuration model
  - \* or a single data node or data tree could be made ephemeral.
5. The management protocol (NETCONF/RESTCONF) needs to signal which portions of a data model (node, tree, or data model) are ephemeral in the module library [RFC7895].

#### 5. IANA Considerations

This is a protocol strawman - nothing is going to IANA.

#### 6. Security Considerations

The security requirements for the I2RS protocol are covered in [I-D.ietf-i2rs-protocol-security-requirements]. The security environment the I2RS protocol is covered in [I-D.ietf-i2rs-security-environment-reqs]. Any person implementing

or deploying the I2RS protocol should consider both security requirements.

## 7. Acknowledgements

TBD

## 8. References

### 8.1. 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, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC4107] Bellovin, S. and R. Housley, "Guidelines for Cryptographic Key Management", BCP 107, RFC 4107, DOI 10.17487/RFC4107, June 2005, <<http://www.rfc-editor.org/info/rfc4107>>.
- [RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <<http://www.rfc-editor.org/info/rfc4960>>.
- [RFC5339] Le Roux, J.L., Ed. and D. Papadimitriou, Ed., "Evaluation of Existing GMPLS Protocols against Multi-Layer and Multi-Region Networks (MLN/MRN)", RFC 5339, DOI 10.17487/RFC5339, September 2008, <<http://www.rfc-editor.org/info/rfc5339>>.
- [RFC5424] Gerhards, R., "The Syslog Protocol", RFC 5424, DOI 10.17487/RFC5424, March 2009, <<http://www.rfc-editor.org/info/rfc5424>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<http://www.rfc-editor.org/info/rfc6020>>.
- [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, <<http://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, <<http://www.rfc-editor.org/info/rfc6242>>.

- [RFC6244] Shafer, P., "An Architecture for Network Management Using NETCONF and YANG", RFC 6244, DOI 10.17487/RFC6244, June 2011, <<http://www.rfc-editor.org/info/rfc6244>>.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <<http://www.rfc-editor.org/info/rfc6536>>.
- [RFC7158] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7158, DOI 10.17487/RFC7158, March 2014, <<http://www.rfc-editor.org/info/rfc7158>>.
- [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, <<http://www.rfc-editor.org/info/rfc7589>>.
- [RFC7803] Leiba, B., "Changing the Registration Policy for the NETCONF Capability URNs Registry", BCP 203, RFC 7803, DOI 10.17487/RFC7803, February 2016, <<http://www.rfc-editor.org/info/rfc7803>>.
- [RFC7895] Bierman, A., Bjorklund, M., and K. Watsen, "YANG Module Library", RFC 7895, DOI 10.17487/RFC7895, June 2016, <<http://www.rfc-editor.org/info/rfc7895>>.
- [RFC7920] Atlas, A., Ed., Nadeau, T., Ed., and D. Ward, "Problem Statement for the Interface to the Routing System", RFC 7920, DOI 10.17487/RFC7920, June 2016, <<http://www.rfc-editor.org/info/rfc7920>>.
- [RFC7921] Atlas, A., Halpern, J., Hares, S., Ward, D., and T. Nadeau, "An Architecture for the Interface to the Routing System", RFC 7921, DOI 10.17487/RFC7921, June 2016, <<http://www.rfc-editor.org/info/rfc7921>>.
- [RFC7922] Clarke, J., Salgueiro, G., and C. Pignataro, "Interface to the Routing System (I2RS) Traceability: Framework and Information Model", RFC 7922, DOI 10.17487/RFC7922, June 2016, <<http://www.rfc-editor.org/info/rfc7922>>.
- [RFC7923] Voit, E., Clemm, A., and A. Gonzalez Prieto, "Requirements for Subscription to YANG Datastores", RFC 7923, DOI 10.17487/RFC7923, June 2016, <<http://www.rfc-editor.org/info/rfc7923>>.

- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<http://www.rfc-editor.org/info/rfc7950>>.
- [RFC7952] Lhotka, L., "Defining and Using Metadata with YANG", RFC 7952, DOI 10.17487/RFC7952, August 2016, <<http://www.rfc-editor.org/info/rfc7952>>.
- [RFC7958] Abley, J., Schlyter, J., Bailey, G., and P. Hoffman, "DNSSEC Trust Anchor Publication for the Root Zone", RFC 7958, DOI 10.17487/RFC7958, August 2016, <<http://www.rfc-editor.org/info/rfc7958>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<http://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, <<http://www.rfc-editor.org/info/rfc8072>>.

## 8.2. Informative References

- [I-D.hares-netconf-i2rs-protocol]  
Hares, S. and a. amit.dass@ericsson.com, "NETCONF Changes to Support I2RS Protocol", draft-hares-netconf-i2rs-protocol-00 (work in progress), November 2016.
- [I-D.hares-netmod-i2rs-yang]  
Hares, S. and a. amit.dass@ericsson.com, "Yang for I2RS Protocol", draft-hares-netmod-i2rs-yang-04 (work in progress), March 2017.
- [I-D.ietf-i2rs-ephemeral-state]  
Haas, J. and S. Hares, "I2RS Ephemeral State Requirements", draft-ietf-i2rs-ephemeral-state-23 (work in progress), November 2016.
- [I-D.ietf-i2rs-protocol-security-requirements]  
Hares, S., Migault, D., and J. Halpern, "I2RS Security Related Requirements", draft-ietf-i2rs-protocol-security-requirements-17 (work in progress), September 2016.



[I-D.ietf-i2rs-rib-data-model]

Wang, L., Ananthakrishnan, H., Chen, M., amit.dass@ericsson.com, a., Kini, S., and N. Bahadur, "A YANG Data Model for Routing Information Base (RIB)", draft-ietf-i2rs-rib-data-model-07 (work in progress), January 2017.

[I-D.ietf-i2rs-rib-info-model]

Bahadur, N., Kini, S., and J. Medved, "Routing Information Base Info Model", draft-ietf-i2rs-rib-info-model-10 (work in progress), December 2016.

[I-D.ietf-i2rs-security-environment-reqs]

Migault, D., Halpern, J., and S. Hares, "I2RS Environment Security Requirements", draft-ietf-i2rs-security-environment-reqs-04 (work in progress), March 2017.

[I-D.ietf-i2rs-yang-l3-topology]

Clemm, A., Medved, J., Varga, R., Liu, X., Ananthakrishnan, H., and N. Bahadur, "A YANG Data Model for Layer 3 Topologies", draft-ietf-i2rs-yang-l3-topology-08 (work in progress), January 2017.

[I-D.ietf-netconf-call-home]

Watsen, K., "NETCONF Call Home and RESTCONF Call Home", draft-ietf-netconf-call-home-17 (work in progress), December 2015.

[I-D.ietf-netconf-keystore]

Watsen, K. and G. Wu, "Keystore Model", draft-ietf-netconf-keystore-00 (work in progress), October 2016.

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

Prieto, A., Clemm, A., Voit, E., Nilsen-Nygaard, E., Tripathy, A., Chisholm, S., and H. Trevino, "NETCONF Support for Event Notifications", draft-ietf-netconf-netconf-event-notifications-01 (work in progress), October 2016.

[I-D.ietf-netconf-restconf]

Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf-18 (work in progress), October 2016.

- [I-D.ietf-netconf-restconf-notif]  
Voit, E., Prieto, A., Tripathy, A., Nilsen-Nygaard, E., Clemm, A., and A. Bierman, "Restconf and HTTP Transport for Event Notifications", draft-ietf-netconf-restconf-notif-02 (work in progress), March 2017.
- [I-D.ietf-netconf-rfc5277bis]  
Clemm, A., Prieto, A., Voit, E., Nilsen-Nygaard, E., Tripathy, A., Chisholm, S., and H. Trevino, "Subscribing to Event Notifications", draft-ietf-netconf-rfc5277bis-01 (work in progress), October 2016.
- [I-D.ietf-netconf-tls-client-server]  
Watsen, K., "TLS Client and Server Models", draft-ietf-netconf-tls-client-server-01 (work in progress), November 2016.
- [I-D.ietf-netconf-yang-patch]  
Bierman, A., Bjorklund, M., and K. Watsen, "YANG Patch Media Type", draft-ietf-netconf-yang-patch-14 (work in progress), November 2016.
- [I-D.ietf-netconf-yang-push]  
Clemm, A., Voit, E., Prieto, A., Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "Subscribing to YANG datastore push updates", draft-ietf-netconf-yang-push-05 (work in progress), March 2017.
- [I-D.ietf-netconf-zerotouch]  
Watsen, K. and M. Abrahamsson, "Zero Touch Provisioning for NETCONF or RESTCONF based Management", draft-ietf-netconf-zerotouch-13 (work in progress), March 2017.
- [I-D.ietf-netmod-revised-datastores]  
Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture", draft-ietf-netmod-revised-datastores-01 (work in progress), March 2017.
- [I-D.ietf-netmod-schema-mount]  
Bjorklund, M. and L. Lhotka, "YANG Schema Mount", draft-ietf-netmod-schema-mount-04 (work in progress), March 2017.
- [I-D.ietf-netmod-syslog-model]  
Wildes, C. and K. Koushik, "A YANG Data Model for Syslog Configuration", draft-ietf-netmod-syslog-model-13 (work in progress), March 2017.

Authors' Addresses

Susan Hares  
Huawei  
Saline  
US

Email: [shares@ndzh.com](mailto:shares@ndzh.com)

Amit Daas  
Ericsson

Email: [amit.dass@ericsson.com](mailto:amit.dass@ericsson.com)

I2RS working group  
Internet-Draft  
Intended status: Informational  
Expires: September 30, 2017

S. Hares  
Huawei  
A. Dass  
Ericsson  
March 29, 2017

RESTCONF Changes to Support I2RS Protocol  
draft-hares-netconf-i2rs-restconf-02.txt

Abstract

This document describes two RESTCONF optional capabilities (i2rs-control plane capability, ephemeral state capabilities) that are needed to support the I2RS protocol needs.

The purpose of this draft is to kick-start the discussions with I2RS Working Group and NETCONF WG on these two capabilities.

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 <http://datatracker.ietf.org/drafts/current/>.

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

This Internet-Draft will expire on September 30, 2017.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
1.1. Background on I2RS . . . . .	3
1.2. Structure of draft . . . . .	3
2. Definitions and Background on I2RS . . . . .	3
2.1. IETF Requirements language . . . . .	3
2.2. I2RS Definitions . . . . .	3
2.3. I2RS protocol requirements . . . . .	5
3. RESTCONF control plane datastore capability . . . . .	5
3.1. Overview . . . . .	5
3.2. Dependencies . . . . .	6
3.3. New Operations . . . . .	6
3.4. Modified Operations . . . . .	6
4. RESTCONF protocol extensions for the ephemeral datastore . .	6
4.1. Overview . . . . .	6
4.2. Dependencies . . . . .	7
4.3. Capability identifier . . . . .	8
4.4. New Operations . . . . .	8
4.5. Modification to data resources . . . . .	8
4.6. Modification to existing operations . . . . .	8
5. IANA Considerations . . . . .	9
6. Security Considerations . . . . .	9
7. Acknowledgements . . . . .	9
8. References . . . . .	9
8.1. Normative References: . . . . .	9
8.2. Informative References . . . . .	11
Authors' Addresses . . . . .	14

## 1. Introduction

This a proposal for the following two RESTCONF capabilities to augment RESTCONF [RFC8040] to support the first version of the I2RS protocol: Control plane datstore capability and ephemeral state capability. The yang that supports this proposal is described in [I-D.hares-netmod-i2rs-yang]. This work is based on the datastore definitions in [I-D.ietf-netmod-revised-datastores].

This draft parallels a similar proposal for NETCONF [RFC6241] is described in [I-D.hares-netconf-i2rs-protocol]. One difference between the proposed capabilities for i2rs control-plane capability additions to NETCONF and the proposed capabilities for i2rs control-plane for RESTCONF is write-collection. RESTCONF has edit-collision capability already which only needs a usage description.

### 1.1. Background on I2RS

The I2RS architecture [RFC7921] defines the I2RS interface "a programmatic interface for state transfer in and out of the Internet routing system". The I2RS protocol is a protocol designed to a higher level protocol comprised of a set of existing protocols which have been extended to work together to support a new interface to the routing system. The I2RS protocol is a "reuse" management protocol which creates new management protocols by reusing existing protocols and extending these protocols for new uses, and has been designed to be implemented in phases [RFC7921].

### 1.2. Structure of draft

The structure of this document is:

Section 2 provides definitions and background on I2RS work. (If you are familiar with the I2RS architecture and requirements, you can skip this section.)

Section 3 describes the RESTCONF control plane datastore capability.

Section 4 describes the RESTCONF ephemeral state capability. .

## 2. Definitions and Background on I2RS

This section reviews definitions from I2RS architecture, and provides background on I2RS work for the reader.

### 2.1. IETF 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].

### 2.2. I2RS Definitions

The I2RS architecture [RFC7921] defines the following terms:

ephemeral data: is data which does not persist across a reboot (software or hardware) or a power on/off condition. Ephemeral data can be configured data or data recorded from operations of the router. Ephemeral configuration data also has the property that a system cannot roll back to a previous ephemeral configuration state. (See [RFC7921] for an architectural overview, [I-D.ietf-i2rs-ephemeral-state] for requirements, and [I-D.ietf-netmod-revised-datastores] for discussion of how the

ephemeral datastore as a control plane datastore interacts with intended datastore and dynamic configuration protocols to form the applied datastore".

local configuration: is the data on a routing system which does persist across a reboot (software or hardware) and a power on/off condition. Local configuration has the ability to roll back to a pervious configuration state. Local configuration is defined as the intended datastore [I-D.ietf-netmod-revised-datastores] which is modified by dynamic configuration protocols (such as DHCP) and the I2RS ephemeral data store

dynamic configuration protocols datastore are configuration protocols such as DHCP that interact with the intended datastore (which does persist across a reboot (software or hardware) power on/off condition), and the I2RS ephemeral state control plane datastore.

control plane protocols datastore is a datastore which is loaded by control plane protocols (e.g. I2RS protocol) rather than system configuration protocols. (see [I-D.ietf-netmod-revised-datastores]).

operator-applied policy: is a policy that an operator sets that determines how the ephemeral datastore as a control plane data store interacts with applied datastore (as defined in [I-D.ietf-netmod-revised-datastores]). This operator policy consists of policy knobs that the operator sets to determine how the I2RS agent control plane ephemeral state datastore will interact with the intended configuration datastor and the dynamic configuration protocol datastore. Three policy knobs could be used to implement this policy:

- \* policy knob 1: I2RS Ephemeral control-plane datastore takes takes precedence over the intended datastore in the routing protocols.
- \* policy knob 2: Updated intended configuration datastore takes precedence over the I2RS ephemeral control-plane data store in the routing protocols
- \* policy knob 3: Ephemeral control plane datastore takes precedence over any other dynamic configuration protocols datastore.

### 2.3. I2RS protocol requirements

The requirements for the I2RS protocol are defined in the following documents:

- o I2RS Problem Statement [RFC7920],
- o I2RS Architecture [RFC7921],
- o I2RS Traceability [RFC7922],
- o Publication and Subscription [RFC7923],
- o I2RS Ephemeral State Requirements, ,  
[I-D.ietf-i2rs-ephemeral-state]
- o I2RS Protocol Security Requirements,  
[I-D.ietf-i2rs-protocol-security-requirements]

The Interface to the routing System (I2RS) creates a new capability for the routing systems, and with greater capabilities come a greater need for security. The requirements for a secure environment for I2RS is described in [I-D.ietf-i2rs-security-environment-reqs].

### 3. RESTCONF control plane datastore capability

capability-name: i2rs-control-plane

#### 3.1. Overview

The i2rs-control-plane datastore capability enables the RESTCONF to support the following dynamic control plane datastore.

- o API resource that is {+restconf}/datastore/<datastore-name>/data/ and operational state specific to the control plane datastore ({+restconf/cp-data/opstate}).
- o It also includes the ability to have the applied datastore and the opstate datastore (per [I-D.ietf-netmod-revised-datastores]) with the ability to return meta-data with the following information:
  - \* Entity-Tag encoding of <client-id><priority> or any portion of the filter.
  - \* "with defaults"
  - \* "with validation" - Yang specified validation (Unclear if this is the best way for validation.)



Ability to provide read access for the configuration datastore

Ability to provide read access for other dynamic datastores

### 3.2. Dependencies

This protocol strawman utilizes the following existing proposed features for NETCONF and RESTCONF

- o RESTCONF [RFC8040].
- o Module library [RFC7895],
- o RESTCONF Patch Media Type [RFC8072],
- o NETCONF Support for event notifications [I-D.ietf-netconf-netconf-event-notifications],
- o Publication/Subscription via Push [I-D.ietf-netconf-yang-push],
- o NETCONF and HTTP Transport for Event Notifications [I-D.ietf-netconf-restconf-notif],
- o Publication/Subscription via Push [I-D.ietf-netconf-yang-push],
- o syslog yang module (both [RFC5424] and [I-D.ietf-netmod-syslog-model])

### 3.3. New Operations

none

### 3.4. Modified Operations

All RESTCONF methods (OPTIONS, HEAD, GET, POST, PUT, PATCH, DELETE) need to work in the control plane datastores. config=TRUE data, and where appropriate config=FALSE data.

## 4. RESTCONF protocol extensions for the ephemeral datastore

capability-name: ephemeral-state

### 4.1. Overview

This capability defines the RESTCONF protocol extensions for control plane protocols that support control plane data stores with ephemeral data.

Ephemeral state is not unique to I2RS work.

The ephemeral capability is the ability to support a dynamic datastores which are entirely ephemeral or have ephemeral state modules, or ephemeral statements within objects in a modules. These objects can be configuration state (config=TRUE) or operational state (config=FALSE).

Ephemeral state in datastores, ephemeral modules or ephemeral objects within a module have one key characteristics: the data does not persist across reboots. The ephemeral configuration state must be restored by a client, and the operational state will need to be regenerated.

The entire requirements for ephemeral state for the I2RS control plane protocol are listed in [I-D.ietf-i2rs-ephemeral-state]. Compared to RESTCONF functionality there are 4 groups of additional changes:

**Constraints** The ability to enforce the constraints for get (aka read) references (to/from) the {+restconf/data} datastore, and {+restconf/cp-data} control plane datastore. ((see Ephemeral-REQ-02, Ephemeral-REQ-03, and Ephemeral-REQ-04 in [I-D.ietf-i2rs-ephemeral-state])). The "validation" yang statement in [I-D.hares-netmod-i2rs-yang] could encode specific validation for the ephemeral case per datastore or per object. [Editor's note: Aid is needed to determine how validation occurs.]

**Ephemeral in Data Modules** Yang modules must identify Yang objects (modules, submodules or objects within yang modules which are ephemeral and augment other nodes) and allow an "ephemeral=TRUE" feature.

**Roll-back** an ephemeral node cannot roll-back to its previous value,

#### 4.2. Dependencies

The ephemeral capabilities have the following dependencies:

- o Yang modules must support the following:
  - \* identifying datastores, modules, and objects as ephemeral. (ephemeral=True)
  - \* Ability to have control plane datastores which are ephemeral.
- o The following features must be supported by RESTCONF

- \* Module library [RFC7895],
- \* RESTCONF Protocol [RFC8040],
- \* RESTCONF Patch Media Type [RFC8072],
- \* NETCONF Support for event notifications [I-D.ietf-netconf-netconf-event-notifications],
- \* Publication/Subscription via Push [I-D.ietf-netconf-yang-push],
- \* NETCONF and HTTP Transport for Event Notifications [I-D.ietf-netconf-restconf-notif],
- \* Subscribing to Yang datastore push updates [I-D.ietf-netconf-yang-push],

#### 4.3. Capability identifier

The ephemeral-datastore capability is identified by the following capability string: ephemeral (TBD URI)

#### 4.4. New Operations

none

#### 4.5. Modification to data resources

RESTCONF must be able to support the ephemeral data in an an control-plane dynamic datastore. This is any API resource that is `{+restconf}/datastore/<datastore-name>/data/` and operational state specific to the control plane datastore (`{+restconf/cp-data/opstate}`).

RESTCONF library functions must be able to store an indication that a data module has ephemeral state as meta-data.

#### 4.6. Modification to existing operations

RESTCONF operations of GET, POST, PUT, PATCH, and DELETE must be able to filter on meta-data with "ephemeral" flag. (Should this be only read).

The operations must support the following things about ephemeral.

1. The ephemeral does not persist over a reboot,
2. an ephemeral node cannot roll-back to its previous value,

## 5. IANA Considerations

TBD -

## 6. Security Considerations

The security requirements for the I2RS protocol are covered in [I-D.ietf-i2rs-protocol-security-requirements]. The security environment the I2RS protocol is covered in [I-D.ietf-i2rs-security-environment-reqs]. Any person implementing or deploying the I2RS protocol should consider both security requirements.

## 7. Acknowledgements

TBD

## 8. References

### 8.1. 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, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC4107] Bellovin, S. and R. Housley, "Guidelines for Cryptographic Key Management", BCP 107, RFC 4107, DOI 10.17487/RFC4107, June 2005, <<http://www.rfc-editor.org/info/rfc4107>>.
- [RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <<http://www.rfc-editor.org/info/rfc4960>>.
- [RFC5339] Le Roux, JL., Ed. and D. Papadimitriou, Ed., "Evaluation of Existing GMPLS Protocols against Multi-Layer and Multi-Region Networks (MLN/MRN)", RFC 5339, DOI 10.17487/RFC5339, September 2008, <<http://www.rfc-editor.org/info/rfc5339>>.
- [RFC5424] Gerhards, R., "The Syslog Protocol", RFC 5424, DOI 10.17487/RFC5424, March 2009, <<http://www.rfc-editor.org/info/rfc5424>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<http://www.rfc-editor.org/info/rfc6020>>.

- [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, <<http://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, <<http://www.rfc-editor.org/info/rfc6242>>.
- [RFC6244] Shafer, P., "An Architecture for Network Management Using NETCONF and YANG", RFC 6244, DOI 10.17487/RFC6244, June 2011, <<http://www.rfc-editor.org/info/rfc6244>>.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <<http://www.rfc-editor.org/info/rfc6536>>.
- [RFC7158] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7158, DOI 10.17487/RFC7158, March 2014, <<http://www.rfc-editor.org/info/rfc7158>>.
- [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, <<http://www.rfc-editor.org/info/rfc7589>>.
- [RFC7803] Leiba, B., "Changing the Registration Policy for the NETCONF Capability URNs Registry", BCP 203, RFC 7803, DOI 10.17487/RFC7803, February 2016, <<http://www.rfc-editor.org/info/rfc7803>>.
- [RFC7895] Bierman, A., Bjorklund, M., and K. Watsen, "YANG Module Library", RFC 7895, DOI 10.17487/RFC7895, June 2016, <<http://www.rfc-editor.org/info/rfc7895>>.
- [RFC7920] Atlas, A., Ed., Nadeau, T., Ed., and D. Ward, "Problem Statement for the Interface to the Routing System", RFC 7920, DOI 10.17487/RFC7920, June 2016, <<http://www.rfc-editor.org/info/rfc7920>>.
- [RFC7921] Atlas, A., Halpern, J., Hares, S., Ward, D., and T. Nadeau, "An Architecture for the Interface to the Routing System", RFC 7921, DOI 10.17487/RFC7921, June 2016, <<http://www.rfc-editor.org/info/rfc7921>>.

- [RFC7922] Clarke, J., Salgueiro, G., and C. Pignataro, "Interface to the Routing System (I2RS) Traceability: Framework and Information Model", RFC 7922, DOI 10.17487/RFC7922, June 2016, <<http://www.rfc-editor.org/info/rfc7922>>.
- [RFC7923] Voit, E., Clemm, A., and A. Gonzalez Prieto, "Requirements for Subscription to YANG Datastores", RFC 7923, DOI 10.17487/RFC7923, June 2016, <<http://www.rfc-editor.org/info/rfc7923>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<http://www.rfc-editor.org/info/rfc7950>>.
- [RFC7952] Lhotka, L., "Defining and Using Metadata with YANG", RFC 7952, DOI 10.17487/RFC7952, August 2016, <<http://www.rfc-editor.org/info/rfc7952>>.
- [RFC7958] Abley, J., Schlyter, J., Bailey, G., and P. Hoffman, "DNSSEC Trust Anchor Publication for the Root Zone", RFC 7958, DOI 10.17487/RFC7958, August 2016, <<http://www.rfc-editor.org/info/rfc7958>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<http://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, <<http://www.rfc-editor.org/info/rfc8072>>.

## 8.2. Informative References

- [I-D.hares-netconf-i2rs-protocol]  
Hares, S. and a. amit.dass@ericsson.com, "NETCONF Changes to Support I2RS Protocol", draft-hares-netconf-i2rs-protocol-00 (work in progress), November 2016.
- [I-D.hares-netmod-i2rs-yang]  
Hares, S. and a. amit.dass@ericsson.com, "Yang for I2RS Protocol", draft-hares-netmod-i2rs-yang-04 (work in progress), March 2017.
- [I-D.ietf-i2rs-ephemeral-state]  
Haas, J. and S. Hares, "I2RS Ephemeral State Requirements", draft-ietf-i2rs-ephemeral-state-23 (work in progress), November 2016.

- [I-D.ietf-i2rs-protocol-security-requirements]  
Hares, S., Migault, D., and J. Halpern, "I2RS Security Related Requirements", draft-ietf-i2rs-protocol-security-requirements-17 (work in progress), September 2016.
- [I-D.ietf-i2rs-rib-data-model]  
Wang, L., Ananthakrishnan, H., Chen, M., amit.dass@ericsson.com, a., Kini, S., and N. Bahadur, "A YANG Data Model for Routing Information Base (RIB)", draft-ietf-i2rs-rib-data-model-07 (work in progress), January 2017.
- [I-D.ietf-i2rs-rib-info-model]  
Bahadur, N., Kini, S., and J. Medved, "Routing Information Base Info Model", draft-ietf-i2rs-rib-info-model-10 (work in progress), December 2016.
- [I-D.ietf-i2rs-security-environment-reqs]  
Migault, D., Halpern, J., and S. Hares, "I2RS Environment Security Requirements", draft-ietf-i2rs-security-environment-reqs-05 (work in progress), March 2017.
- [I-D.ietf-i2rs-yang-l3-topology]  
Clemm, A., Medved, J., Varga, R., Liu, X., Ananthakrishnan, H., and N. Bahadur, "A YANG Data Model for Layer 3 Topologies", draft-ietf-i2rs-yang-l3-topology-08 (work in progress), January 2017.
- [I-D.ietf-netconf-call-home]  
Watsen, K., "NETCONF Call Home and RESTCONF Call Home", draft-ietf-netconf-call-home-17 (work in progress), December 2015.
- [I-D.ietf-netconf-keystore]  
Watsen, K., "Keystore Model", draft-ietf-netconf-keystore-01 (work in progress), March 2017.
- [I-D.ietf-netconf-netconf-event-notifications]  
Prieto, A., Clemm, A., Voit, E., Nilsen-Nygaard, E., Tripathy, A., Chisholm, S., and H. Trevino, "NETCONF Support for Event Notifications", draft-ietf-netconf-netconf-event-notifications-01 (work in progress), October 2016.
- [I-D.ietf-netconf-restconf]  
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf-18 (work in progress), October 2016.

- [I-D.ietf-netconf-restconf-notif]  
Voit, E., Prieto, A., Tripathy, A., Nilsen-Nygaard, E., Clemm, A., and A. Bierman, "Restconf and HTTP Transport for Event Notifications", draft-ietf-netconf-restconf-notif-02 (work in progress), March 2017.
- [I-D.ietf-netconf-rfc5277bis]  
Clemm, A., Prieto, A., Voit, E., Nilsen-Nygaard, E., Tripathy, A., Chisholm, S., and H. Trevino, "Subscribing to Event Notifications", draft-ietf-netconf-rfc5277bis-01 (work in progress), October 2016.
- [I-D.ietf-netconf-tls-client-server]  
Watsen, K. and G. Wu, "TLS Client and Server Models", draft-ietf-netconf-tls-client-server-02 (work in progress), March 2017.
- [I-D.ietf-netconf-yang-patch]  
Bierman, A., Bjorklund, M., and K. Watsen, "YANG Patch Media Type", draft-ietf-netconf-yang-patch-14 (work in progress), November 2016.
- [I-D.ietf-netconf-yang-push]  
Clemm, A., Voit, E., Prieto, A., Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "Subscribing to YANG datastore push updates", draft-ietf-netconf-yang-push-05 (work in progress), March 2017.
- [I-D.ietf-netconf-zerotouch]  
Watsen, K. and M. Abrahamsson, "Zero Touch Provisioning for NETCONF or RESTCONF based Management", draft-ietf-netconf-zerotouch-13 (work in progress), March 2017.
- [I-D.ietf-netmod-revised-datastores]  
Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture", draft-ietf-netmod-revised-datastores-01 (work in progress), March 2017.
- [I-D.ietf-netmod-schema-mount]  
Bjorklund, M. and L. Lhotka, "YANG Schema Mount", draft-ietf-netmod-schema-mount-04 (work in progress), March 2017.
- [I-D.ietf-netmod-syslog-model]  
Wildes, C. and K. Koushik, "A YANG Data Model for Syslog Configuration", draft-ietf-netmod-syslog-model-14 (work in progress), March 2017.



Authors' Addresses

Susan Hares  
Huawei  
Saline  
US

Email: [shares@ndzh.com](mailto:shares@ndzh.com)

Amit Daas  
Ericsson

Email: [amit.dass@ericsson.com](mailto:amit.dass@ericsson.com)

NETCONF Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: September 14, 2017

K. Watsen  
Juniper Networks  
March 13, 2017

Keystore Model  
draft-ietf-netconf-keystore-01

Abstract

This document defines a YANG data module for a system-level keystore mechanism, that might be used to hold onto private keys and certificates that are trusted by the system advertising support for this module.

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 "2017-03-13" --> the publication date of this draft

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

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

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 <http://datatracker.ietf.org/drafts/current/>.

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

This Internet-Draft will expire on September 14, 2017.

#### Copyright Notice

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

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

#### Table of Contents

1. Introduction . . . . .	3
1.1. Requirements Language . . . . .	3
1.2. Tree Diagram Notation . . . . .	3
2. The Keystore Model . . . . .	4
2.1. Overview . . . . .	4
2.2. Example Usage . . . . .	5
2.3. YANG Module . . . . .	10
3. Design Considerations . . . . .	20
4. Security Considerations . . . . .	21
5. IANA Considerations . . . . .	22
5.1. The IETF XML Registry . . . . .	22
5.2. The YANG Module Names Registry . . . . .	22
6. Acknowledgements . . . . .	23
7. References . . . . .	23
7.1. Normative References . . . . .	23
7.2. Informative References . . . . .	23
Appendix A. Change Log . . . . .	25
A.1. server-model-09 to 00 . . . . .	25
A.2. keychain-00 to keystore-00 . . . . .	25
A.3. 00 to 01 . . . . .	25
Appendix B. Open Issues . . . . .	25

Author's Address . . . . . 25

## 1. Introduction

This document defines a YANG [RFC6020] data module for a system-level keystore mechanism, which can be used to hold onto private keys and certificates that are trusted by the system advertising support for this module.

This module provides a centralized location for security sensitive data, so that the data can be then referenced by other modules. There are two types of data that are maintained by this module:

- o Private keys, and any associated public certificates.
- o Sets of trusted certificates.

This document extends special consideration for systems that have Trusted Protection Modules (TPMs). These systems are unique in that the TPM must be directed to generate new private keys (it is not possible to load a private key into a TPM) and it is not possible to backup/restore the TPM's private keys as configuration.

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

### 1.1. Requirements Language

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

### 1.2. Tree Diagram Notation

A simplified graphical representation of the data models is used in this document. The meaning of the symbols in these diagrams is as follows:

- o Brackets "[" and "]" enclose list keys.
- o Braces "{" and "}" enclose feature names, and indicate that the named feature must be present for the subtree to be present.
- o Abbreviations before data node names: "rw" means configuration (read-write) and "ro" state data (read-only).
- o Symbols after data node names: "?" means an optional node, "!" means a presence container, and "\*" denotes a list and leaf-list.

- o Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- o Ellipsis ("...") stands for contents of subtrees that are not shown.

## 2. The Keystore Model

The keystore module defined in this section provides a configurable object having the following characteristics:

- o A semi-configurable list of private keys, each with one or more associated certificates. Private keys **MUST** be either preinstalled (e.g., a key associated to an IDevID [Std-802.1AR-2009] certificate), be generated by request, or be loaded by request. Each private key is **MAY** have associated certificates, either preinstalled or configured after creation.
- o A configurable list of lists of trust anchor certificates. This enables the server to have use-case specific trust anchors. For instance, one list of trust anchors might be used to authenticate management connections (e.g., client certificate-based authentication for NETCONF or RESTCONF connections), and a different list of trust anchors might be used for when connecting to a specific Internet-based service (e.g., a zero touch bootstrap server).
- o An RPC to generate a certificate signing request for an existing private key, a passed subject, and an optional attributes. The signed certificate returned from an external certificate authority (CA) can be later set using a standard configuration change request (e.g., <edit-config>).
- o An RPC to request the server to generate a new private key using the specified algorithm and key length.
- o An RPC to request the server to load a new private key.

### 2.1. Overview

The keystore module has the following tree diagram. Please see Section 1.2 for information on how to interpret this diagram.

```

module: ietf-keystore
  +--rw keystore
    +--rw keys
      +--rw key* [name]
        +--rw name string
        +--rw algorithm-identifier identityref
        +--rw private-key union
        +--ro public-key binary
        +--rw certificates
          +--rw certificate* [name]
            +--rw name string
            +--rw value? binary
          +---x generate-certificate-signing-request
            +---w input
              +---w subject binary
              +---w attributes? binary
            +--ro output
              +--ro certificate-signing-request binary
        +--rw trusted-certificates* [name]
          +--rw name string
          +--rw description? string
          +--rw trusted-certificate* [name]
            +--rw name string
            +--rw certificate? binary
        +--rw trusted-host-keys* [name]
          +--rw name string
          +--rw description? string
          +--rw trusted-host-key* [name]
            +--rw name string
            +--rw host-key binary

  notifications:
    +---n certificate-expiration
      +--ro certificate instance-identifier
      +--ro expiration-date yang:date-and-time

```

## 2.2. Example Usage

The following example illustrates what a fully configured keystore object might look like. The private-key shown below is consistent with the generate-private-key and generate-certificate-signing-request examples above. This example also assumes that the resulting CA-signed certificate has been configured back onto the server. Lastly, this example shows that three lists of trusted certificates having been configured.

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

```
<!-- private keys and associated certificates -->
<keys>
  <key>
    <name>ex-rsa-key</name>
    <algorithm-identifier>rsa1024</algorithm-identifier>
    <private-key>Base64-encoded RSA Private Key</private-key>
    <public-key>Base64-encoded RSA Public Key</public-key>
    <certificates>
      <certificate>
        <name>ex-rsa-cert</name>
        <value>Base64-encoded PKCS#7</value>
      </certificate>
    </certificates>
  </key>

  <key>
    <name>tls-ec-key</name>
    <algorithm-identifier>secp256r1</algorithm-identifier>
    <private-key>Base64-encoded EC Private Key</private-key>
    <public-key>Base64-encoded EC Public Key</public-key>
    <certificates>
      <certificate>
        <name>tls-ec-cert</name>
        <value>Base64-encoded PKCS#7</value>
      </certificate>
    </certificates>
  </key>

  <key>
    <name>tpm-protected-key</name>
    <algorithm-identifier>rsa2048</algorithm-identifier>
    <private-key>Base64-encoded RSA Private Key</private-key>
    <public-key>Base64-encoded RSA Public Key</public-key>
    <certificates>
      <certificate>
        <name>builtin-idevid-cert</name>
        <value>Base64-encoded PKCS#7</value>
      </certificate>
      <certificate>
        <name>my-ldevid-cert</name>
        <value>Base64-encoded PKCS#7</value>
      </certificate>
    </certificates>
  </key>
</keys>

<!-- trusted netconf/restconf client certificates -->
<trusted-certificates>
```

```
<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 trusted CA.
</description>
<trusted-certificate>
  <name>George Jetson</name>
  <certificate>Base64-encoded X.509v3</certificate>
</trusted-certificate>
</trusted-certificates>

<trusted-certificates>
  <name>explicitly-trusted-server-certs</name>
  <description>
    Specific server authentication certificates for explicitly
    trusted servers. These are needed for server certificates
    that are not signed by a trusted CA.
  </description>
  <trusted-certificate>
    <name>Fred Flintstone</name>
    <certificate>Base64-encoded X.509v3</certificate>
  </trusted-certificate>
</trusted-certificates>

<!-- trust anchors (CA certs) for authenticating clients -->
<trusted-certificates>
  <name>deployment-specific-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 configured
    CA certificates.
  </description>
  <trusted-certificate>
    <name>ca.example.com</name>
    <certificate>Base64-encoded X.509v3</certificate>
  </trusted-certificate>
</trusted-certificates>

<!-- trust anchors for random HTTPS servers on Internet -->
<trusted-certificates>
  <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>
```



```

    <trusted-certificate>
      <name>ex-certificate-authority</name>
      <certificate>Base64-encoded X.509v3</certificate>
    </trusted-certificate>
  </trusted-certificates>

  <!-- trusted SSH host keys -->
  <trusted-host-keys>
    <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>
    <trusted-host-key>
      <name>corp-fw1</name>
      <host-key>Base64-encoded OneAsymmetricKey</host-key>
    </trusted-host-key>
  </trusted-host-keys>

</keystore>

```

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">
    <keystore
      xmlns="urn:ietf:params:xml:ns:yang:ietf-keystore">
      <private-keys>
        <private-key>
          <name>ex-key-sect571r1</name>
          <generate-certificate-signing-request>
            <subject>
              cztvaWRoc2RmZ2tqaHNkZmdramRzZnZzZGtmam5idnNvO2R
              manZvO3NkZmJpdmhZGZpbHVidjtvZ2lkZmhidmllbHNlmlO
              Z2aXNiZGZpYmhZG87ZmJvO3NkZ25iO29pLmR6Zgo=
            </subject>
            <attributes>
              bwtakWRoc2RmZ2tqaHNkZmdramRzZnZzZGtmam5idnNvut4
              arnZvO3NkZmJpdmhZGZpbHVidjtvZ2lkZmhidmllbHNkYm
              Z2aXNiZGZpYmhZG87ZmJvO3NkZ25iO29pLmC6Rhp=
            </attributes>
          </generate-certificate-signing-request>
        </private-key>
      </private-keys>
    </keystore>
  </action>
</rpc>

```

```

        </private-key>
    </private-keys>
</keystore>
</action>
</rpc>

```

## RESPONSE

-----

```

<rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <certificate-signing-request
    xmlns="urn:ietf:params:xml:ns:yang:ietf-keystore">
    LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUNrekNDQWZ5Z
    0F3SUJBZ01KQUptRT2t3bGpNK2pjTUEwR0NTcUdTSWl3RFFFQkJR
    VU FNRRFF4Q3pBSk1JnTlYkQkFZVEFsVlRNUkF3RGdZRFZRUUtd
    Fd2RsZUd GdGNHeGxNUk13RVFZRFZRUURFd3BEVWt3Z1NYTnpk
    V1Z5TUI0WApe diRlV4RXpBUk1JnTlZCQU1UQ2tOU1RDQkpjM04x
    WlhJd2daOHdEUUV1 KS29aSWWh2Y04KQVFFQkJRQURnWTBBTUlH
    S1FvR0JBTVXVvZmFPNEV3 El1QWMrQ1RSTkNmc0d6cEw1Um5y
    dXZsOFRicUJTdGZQY3N0Zk1KT1 FaNzlnNlNWVldsMldzaHE1bU
    ViCk1JNnItGNzdjbTAVU25FcFE0TnV bXBDT2YkQWdNQkFBR2p
    nYXZ3Z2Frd0hRWURWUjBPQk1JRUZKY1o2W URiR01PNDB4ajlP
    b3JtREdsRUNCVTFNRL1FHQTFVZApJd1JkTUZ1QU ZKY1o2WURi
    R01PNDB4ajlPb3JtREdsRUNCVTfVVGlrTm1pBME1Rc3d mMKTUE
    0R0ExVWRed0VCL3dRRUF3SUNCREFTQmdOVkhSTUJBZjhFQ0 RBR0
    FRSC9BZ0VBTUEwR0NTcUdTSWl3RFFFQgpCUVVBQTRHQAkFMMmx
    rWmFGNWcyAGR6MVNhZnZPbnBneHA4eG00SHRhbStadHpLazF1S3Bx
    TXp4YXJCbFpDSH1LCKlVbC9GVzRtV1RQSlVDeEtFTE40NEY2Zmk2
    d c4d0tSSElkyWlWl0pGTmlQS0VXSTF4K1IlaDZmazcrQzQ1QXg1
    RWV SWHgZzjdVM2xZTgotLS0tLUVORCBDRVJUSUZJQ0FURS0tLS0t
    Cg==
  </certificate-signing-request>
</rpc-reply>

```

The following example illustrates a "certificate-expiration" notification in XML.

[\'\\' line wrapping added for formatting only]

```
<notification
  xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2016-07-08T00:01:00Z</eventTime>
  <certificate-expiration
    xmlns="urn:ietf:params:xml:ns:yang:ietf-keystore">
    <certificate>/ks:keystore/ks:private-keys/ks:private-key\
      /ks:certificate-chains/ks:certificate-chain/ks:certificate[3]\
    </certificate>
    <expiration-date>2016-08-08T14:18:53-05:00</expiration-date>
  </certificate-expiration>
</notification>
```

### 2.3. YANG Module

This YANG module makes extensive use of data types defined in [RFC5280] and [RFC5958].

<CODE BEGINS> file "ietf-keystore@2017-03-13.yang"

```
module ietf-keystore {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-keystore";
  prefix "ks";

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

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

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

  contact
    "WG Web:  <http://tools.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) 2014 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

```
revision "2017-03-13" {
  description
    "Initial version";
  reference
    "RFC VVVV: NETCONF Server and RESTCONF Server Configuration
      Models";
}

// Identities

identity key-algorithm {
  description
    "Base identity from which all key-algorithms are derived.";
}

identity rsa1024 {
  base key-algorithm;
  description
    "The RSA algorithm using a 1024-bit key.";
  reference
    "RFC3447: Public-Key Cryptography Standards (PKCS) #1:
      RSA Cryptography Specifications Version 2.1.";
}

identity rsa2048 {
  base key-algorithm;
  description
```

```
    "The RSA algorithm using a 2048-bit key.";
  reference
    "RFC3447: Public-Key Cryptography Standards (PKCS) #1:
      RSA Cryptography Specifications Version 2.1.";
}

identity rsa3072 {
  base key-algorithm;
  description
    "The RSA algorithm using a 3072-bit key.";
  reference
    "RFC3447: Public-Key Cryptography Standards (PKCS) #1:
      RSA Cryptography Specifications Version 2.1.";
}

identity rsa4096 {
  base key-algorithm;
  description
    "The RSA algorithm using a 4096-bit key.";
  reference
    "RFC3447: Public-Key Cryptography Standards (PKCS) #1:
      RSA Cryptography Specifications Version 2.1.";
}

identity rsa7680 {
  base key-algorithm;
  description
    "The RSA algorithm using a 7680-bit key.";
  reference
    "RFC3447: Public-Key Cryptography Standards (PKCS) #1:
      RSA Cryptography Specifications Version 2.1.";
}

identity rsa15360 {
  base key-algorithm;
  description
    "The RSA algorithm using a 15360-bit key.";
  reference
    "RFC3447: Public-Key Cryptography Standards (PKCS) #1:
      RSA Cryptography Specifications Version 2.1.";
}

identity secp192r1 {
  base key-algorithm;
  description
    "The secp192r1 algorithm.";
  reference
    "RFC5480:
```

```
        Elliptic Curve Cryptography Subject Public Key Information.";
    }

    identity secp256r1 {
        base key-algorithm;
        description
            "The secp256r1 algorithm.";
        reference
            "RFC5480:
            Elliptic Curve Cryptography Subject Public Key Information.";
    }

    identity secp384r1 {
        base key-algorithm;
        description
            "The secp384r1 algorithm.";
        reference
            "RFC5480:
            Elliptic Curve Cryptography Subject Public Key Information.";
    }

    identity secp521r1 {
        base key-algorithm;
        description
            "The secp521r1 algorithm.";
        reference
            "RFC5480:
            Elliptic Curve Cryptography Subject Public Key Information.";
    }

    // data model

    container keystore {
        nacm:default-deny-write;
        description
            "The keystore contains both active material (e.g., private keys
            and passwords) and passive material (e.g., trust anchors).

            The active material can be used to support either a server (e.g.,
            a TLS/SSH server's private) or a client (a private key used for
            TLS/SSH client-certificate based authentication, or a password
            used for SSH/HTTP-client authentication).

            The passive material can be used to support either a server
            (e.g., client certificates to trust) or clients (e.g., server
            certificates to trust).";

        container keys {
```

```
description
  "A list of keys maintained by the keystore.";
list key {
  key name;
  description
    "A key maintained by the keystore.";
  leaf name {
    type string;
    description
      "An arbitrary name for the key.";
  }
  leaf algorithm-identifier {
    type identityref {
      base "key-algorithm";
    }
    mandatory true;
    description
      "Identifies which algorithm is to be used with the key.
      This value determines how the 'private-key' and 'public-
      key' fields are interpreted.";
      // no params, such as in RFC 5912? (no are set for algs
      // we care about, but what about the future?
  }
  leaf private-key {
    nacm:default-deny-all;
    type union {
      type binary;
      type enumeration {
        enum "RESTRICTED" {
          description
            "The private key is restricted due to access-control.";
        }
        enum "INACCESSIBLE" {
          description
            "The private key is inaccessible due to being protected
            by the cryptographic hardware modules (e.g., a TPM).";
        }
      }
    }
  }
  mandatory true;
  description
    "A binary string that contains the value of the private
    key. The interpretation of the content is defined in the
    registration of the key algorithm. For example, a DSA key
    is an INTEGER, an RSA key is represented as RSAPrivateKey
    as defined in [RFC3447], and an Elliptic Curve Cryptography
    (ECC) key is represented as ECPrivateKey as defined in
    [RFC5915]"; // text lifted from RFC5958
```

```
}

// no key usage (ref: RFC 5912, pg 101 -- too X.509 specific?)

leaf public-key {
    type binary;
    config false;
    mandatory true;
    description
        "A binary string that contains the value of the public
        key. The interpretation of the content is defined in the
        registration of the key algorithm. For example, a DSA key
        is an INTEGER, an RSA key is represented as RSAPublicKey
        as defined in [RFC3447], and an Elliptic Curve Cryptography
        (ECC) key is represented using the 'publicKey' described in
        [RFC5915]";
}
container certificates {
    description
        "Certificates associated with this private key. More
        than one certificate per key is enabled to support,
        for instance, a TPM-protected key that has associated
        both IDevID and LDevID certificates.";
    list certificate {
        key name;
        description
            "A certificate for this private key.";
        leaf name {
            type string;
            description
                "An arbitrary name for the certificate. The name
                must be a unique across all keys, not just within
                this key.";
        }
        leaf value {
            type binary;
            description
                "An unsigned PKCS #7 SignedData structure, as specified
                by Section 9.1 in RFC 2315, containing just certificates
                (no content, signatures, or CRLs), encoded using ASN.1
                distinguished encoding rules (DER), as specified in
                ITU-T X.690.

                This structure contains, in order, the certificate
                itself and all intermediate certificates leading up
                to a trust anchor certificate. The certificate MAY
                optionally include the trust anchor certificate.";
            reference
```



```
"RFC 2315:
  PKCS #7: Cryptographic Message Syntax Version 1.5.
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).";
}
}
}
action generate-certificate-signing-request {
  description
    "Generates a certificate signing request structure for
    the associated private key using the passed subject and
    attribute values. Please review both the Security
    Considerations and Design Considerations sections in
    RFC VVVV for more information regarding this action
    statement.";
  input {
    leaf subject {
      type binary;
      mandatory true;
      description
        "The 'subject' field from 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 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).";
    }
}
output {
    leaf certificate-signing-request {
        type binary;
        mandatory true;
        description
            "A CertificationRequest 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).";
    }
}
}
}
}

list trusted-certificates {
    key name;
    description
        "A list of trusted certificates. These certificates
        can be used by a server to authenticate clients, or by
        clients to authenticate servers. The certificates may
        be endpoint specific or for certificate authorities,
        to authenticate many clients at once. Each list of
        certificates SHOULD be specific to a purpose, as the
        list as a whole may be referenced by other modules.
        For instance, a NETCONF server model might point to
        a list of certificates to use when authenticating
        client certificates.";
    leaf name {
        type string;
        description
            "An arbitrary name for this list of trusted certificates.";
    }
}

```

```

    }
    leaf description {
        type string;
        description
            "An arbitrary description for this list of trusted
            certificates.";
    }
    list trusted-certificate {
        key name;
        description
            "A trusted certificate for a specific use. Note, this
            'certificate' is a list in order to encode any
            associated intermediate certificates.";
        leaf name {
            type string;
            description
                "An arbitrary name for this trusted certificate. Must
                be unique across all lists of trusted certificates
                (not just this list) so that a leafref to it from
                another module can resolve to unique values.";
        }
        leaf certificate { // rename to 'data'?
            type binary;
            description
                "An X.509 v3 certificate structure as specified by RFC
                5280, Section 4 encoded using the 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).";
        }
    }
}

list trusted-host-keys {
    key name;
    description
        "A list of trusted host-keys. These host-keys can be used
        by clients to authenticate SSH servers. The host-keys are
        endpoint specific. Each list of host-keys SHOULD be
        specific to a purpose, as the list as a whole may be
        referenced by other modules. For instance, a NETCONF

```

```

    client model might point to a list of host-keys to use
    when authenticating servers host-keys.";
leaf name {
    type string;
    description
        "An arbitrary name for this list of trusted SSH host keys.";
}
leaf description {
    type string;
    description
        "An arbitrary description for this list of trusted SSH host
        keys.";
}
list trusted-host-key {
    key name;
    description
        "A trusted host key.";
    leaf name {
        type string;
        description
            "An arbitrary name for this trusted host-key. Must be
            unique across all lists of trusted host-keys (not just
            this list) so that a leafref to it from another module
            can resolve to unique values.

            Note that, for when the SSH client is able to listen
            for call-home connections as well, there is no reference
            identifier (e.g., hostname, IP address, etc.) that it
            can use to uniquely identify the server with. The
            call-home draft recommends SSH servers use X.509v3
            certificates (RFC6187) when calling home.";
    }
}
leaf host-key { // rename to 'data'?
    type binary;
    mandatory true;
    description // is this the correct type?
        "An OneAsymmetricKey 'publicKey' structure as specified
        by RFC 5958, Section 2 encoded using the ASN.1
        distinguished encoding rules (DER), as specified
        in ITU-T X.690.";
    reference
        "RFC 5958:
            Asymmetric Key Packages
            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).";
}

```

```

    }
  }
}

notification certificate-expiration {
  description
    "A notification indicating that a 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.";
  leaf certificate {
    type instance-identifier;
    mandatory true;
    description
      "Identifies which certificate is expiring or is expired.";
  }
  leaf expiration-date {
    type yang:date-and-time;
    mandatory true;
    description
      "Identifies the expiration date on the certificate.";
  }
}
}

```

<CODE ENDS>

### 3. Design Considerations

This document uses PKCS #10 [RFC2986] for the "generate-certificate-signing-request" action. The use of Certificate Request Message Format (CRMF) [RFC4211] was considered, but it was unclear if there was market demand for it, and so support for CRMF has been left out of this specification. 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.

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

For the trusted-certificates list, Trust Anchor Format [RFC5914] was evaluated and deemed inappropriate due to this document's need to also support pinning. That is, pinning a client-certificate to support NETCONF over TLS client authentication.

#### 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) [RFC6536] 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, trusted anchors, etc., can dramatically alter the implemented security policy. This being the case, the top-level node in this module is marked with the NACM value 'default-deny-write'.

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

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or

notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

/keystore/keys/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. This being the case, this node is marked with the NACM value 'default-deny-all'.

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:

generate-certificate-signing-request: For this RPC operation, 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 in the secure transport layer was established.

## 5. IANA Considerations

### 5.1. The IETF XML Registry

This document registers one URI in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following 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: kc  
reference: RFC VVVV

## 6. Acknowledgements

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

## 7. References

### 7.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC2986] Nystrom, M. and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7", RFC 2986, DOI 10.17487/RFC2986, November 2000, <<http://www.rfc-editor.org/info/rfc2986>>.
- [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, <<http://www.rfc-editor.org/info/rfc5280>>.
- [RFC5958] Turner, S., "Asymmetric Key Packages", RFC 5958, DOI 10.17487/RFC5958, August 2010, <<http://www.rfc-editor.org/info/rfc5958>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<http://www.rfc-editor.org/info/rfc6020>>.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <<http://www.rfc-editor.org/info/rfc6536>>.

### 7.2. Informative References

- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<http://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, <<http://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, <<http://www.rfc-editor.org/info/rfc5056>>.
- [RFC5914] Housley, R., Ashmore, S., and C. Wallace, "Trust Anchor Format", RFC 5914, DOI 10.17487/RFC5914, June 2010, <<http://www.rfc-editor.org/info/rfc5914>>.
- [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, <<http://www.rfc-editor.org/info/rfc6241>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<http://www.rfc-editor.org/info/rfc8040>>.
- [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. server-model-09 to 00

- o This draft was split out from draft-ietf-netconf-server-model-09.
- o Removed key-usage parameter from generate-private-key action.
- o Now /private-keys/private-key/certificates/certificate/name must be globally unique (unique across all private keys).
- o Added top-level 'trusted-ssh-host-keys' and 'user-auth-credentials' to support SSH client modules.

## A.2. keychain-00 to keystore-00

- o Renamed module from "keychain" to "keystore" (Issue #3)

## A.3. 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)

## Appendix B. Open Issues

Please see: <https://github.com/netconf-wg/keystore/issues>.

## Author's Address

Kent Watsen  
Juniper Networks

EMail: [kwatsen@juniper.net](mailto:kwatsen@juniper.net)

NETCONF Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: 8 September 2022

K. Watsen  
Watsen Networks  
7 March 2022

A YANG Data Model for a Keystore  
draft-ietf-netconf-keystore-24

Abstract

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

Editorial Note (To be removed by RFC Editor)

This draft contains 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:

- \* AAAA --> the assigned RFC value for draft-ietf-netconf-crypto-types
- \* CCCC --> 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:

- \* 2022-03-07 --> the publication date of this draft

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

- \* Appendix A. Change Log

Status of This Memo

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

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

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

This Internet-Draft will expire on 8 September 2022.

#### Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

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

#### Table of Contents

1. Introduction . . . . .	3
1.1. Relation to other RFCs . . . . .	4
1.2. Specification Language . . . . .	6
1.3. Terminology . . . . .	6
1.4. Adherence to the NMDA . . . . .	6
1.5. Conventions . . . . .	6
2. The "ietf-keystore" Module . . . . .	6
2.1. Data Model Overview . . . . .	7
2.2. Example Usage . . . . .	14
2.3. YANG Module . . . . .	26
3. Support for Built-in Keys . . . . .	35
4. Encrypting Keys in Configuration . . . . .	37
5. Security Considerations . . . . .	41
5.1. Security of Data at Rest . . . . .	41
5.2. Unconstrained Private Key Usage . . . . .	41
5.3. The "ietf-keystore" YANG Module . . . . .	41
6. IANA Considerations . . . . .	42
6.1. The "IETF XML" Registry . . . . .	42
6.2. The "YANG Module Names" Registry . . . . .	42
7. References . . . . .	42

7.1. Normative References . . . . .	42
7.2. Informative References . . . . .	43
Appendix A. Change Log . . . . .	45
A.1. 00 to 01 . . . . .	45
A.2. 01 to 02 . . . . .	45
A.3. 02 to 03 . . . . .	46
A.4. 03 to 04 . . . . .	46
A.5. 04 to 05 . . . . .	46
A.6. 05 to 06 . . . . .	46
A.7. 06 to 07 . . . . .	47
A.8. 07 to 08 . . . . .	47
A.9. 08 to 09 . . . . .	47
A.10. 09 to 10 . . . . .	47
A.11. 10 to 11 . . . . .	48
A.12. 11 to 12 . . . . .	48
A.13. 12 to 13 . . . . .	48
A.14. 13 to 14 . . . . .	48
A.15. 14 to 15 . . . . .	48
A.16. 15 to 16 . . . . .	49
A.17. 16 to 17 . . . . .	49
A.18. 17 to 18 . . . . .	49
A.19. 18 to 19 . . . . .	50
A.20. 19 to 20 . . . . .	50
A.21. 20 to 21 . . . . .	50
A.22. 21 to 22 . . . . .	50
A.23. 22 to 23 . . . . .	50
A.24. 23 to 24 . . . . .	51
Acknowledgements . . . . .	51
Author's Address . . . . .	51

## 1. Introduction

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

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

Special consideration has been given for systems that have cryptographic hardware, such as a Trusted Platform Module (TPM). These systems are unique in that the cryptographic hardware hides the

secret key values. Additionally, such hardware is commonly initialized when manufactured to protect a "built-in" asymmetric key for which the public half is conveyed in an identity certificate (e.g., an IDevID [Std-802.1AR-2018] certificate). Please see Section 3 to see how built-in keys are supported.

This document intends to support existing practices; it does not intend to define new behavior for systems to implement. To simplify implementation, advanced key formats may be selectively implemented.

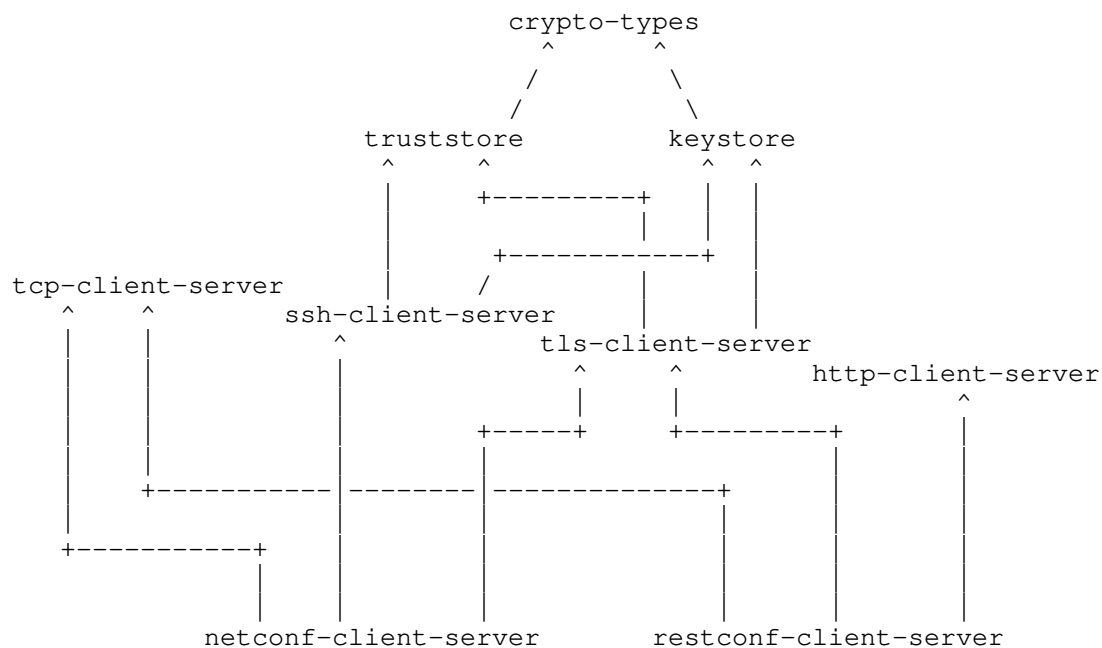
Implementations may utilize zero or more operating system level keystore utilities and/or hardware security modules (HSMs).

### 1.1. Relation to other RFCs

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

The modules have been defined in a modular fashion to enable their use by other efforts, some of which are known to be in progress at the time of this writing, with many more expected to be defined in time.

The normative dependency relationship between the various RFCs in the collection is presented in the below diagram. The labels in the diagram represent the primary purpose provided by each RFC. Hyperlinks to each RFC are provided below the diagram.



Label in Diagram	Originating RFC
crypto-types	[I-D.ietf-netconf-crypto-types]
truststore	[I-D.ietf-netconf-trust-anchors]
keystore	[I-D.ietf-netconf-keystore]
tcp-client-server	[I-D.ietf-netconf-tcp-client-server]
ssh-client-server	[I-D.ietf-netconf-ssh-client-server]
tls-client-server	[I-D.ietf-netconf-tls-client-server]
http-client-server	[I-D.ietf-netconf-http-client-server]
netconf-client-server	[I-D.ietf-netconf-netconf-client-server]
restconf-client-server	[I-D.ietf-netconf-restconf-client-server]

Table 1: Label to RFC Mapping

## 1.2. Specification Language

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

## 1.3. Terminology

The terms "client" and "server" are defined in [RFC6241] and are not redefined here.

The term "keystore" is defined in this draft as a mechanism that intends safeguard secrets placed into it for protection.

The nomenclature "<running>" and "<operational>" are defined in [RFC8342].

The sentence fragments "augmented" and "augmented in" are used herein as the past tense verbified form of the "augment" statement defined in Section 7.17 of [RFC7950].

## 1.4. Adherence to the NMDA

This document is compliant with Network Management Datastore Architecture (NMDA) [RFC8342]. For instance, keys and associated certificates installed during manufacturing (e.g., for an IDevID certificate) are expected to appear in <operational> (see Section 3).

## 1.5. Conventions

Various examples used in this document use a placeholder value for binary data that has been base64 encoded (e.g., "BASE64VALUE="). This placeholder value is used as real base64 encoded structures are often many lines long and hence distracting to the example being presented.

## 2. The "ietf-keystore" Module

This section defines a YANG 1.1 [RFC7950] module called "ietf-keystore". A high-level overview of the module is provided in Section 2.1. Examples illustrating the module's use are provided in Section 2.2. The YANG module itself is defined in Section 2.3.



## 2.1. Data Model Overview

This section provides an overview of the "ietf-keystore" module in terms of its features, typedefs, groupings, and protocol-accessible nodes.

### 2.1.1. Features

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

Features:

```
+-- central-keystore-supported
+-- local-definitions-supported
```

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

### 2.1.2. Typedefs

The following diagram lists the "typedef" statements defined in the "ietf-keystore" module:

Typedefs:

```
leafref
+-- symmetric-key-ref
+-- asymmetric-key-ref
```

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

Comments:

- \* All the typedefs defined in the "ietf-keystore" module extend the base "leafref" type defined in [RFC7950].
- \* The leafrefs refer to symmetric and asymmetric keys in the central keystore, when this module is implemented.
- \* These typedefs are provided as an aid to downstream modules that import the "ietf-keystore" module.

### 2.1.3. Groupings

The "ietf-keystore" module defines the following "grouping" statements:

- \* encrypted-by-choice-grouping

- \* asymmetric-key-certificate-ref-grouping
- \* local-or-keystore-symmetric-key-grouping
- \* local-or-keystore-asymmetric-key-grouping
- \* local-or-keystore-asymmetric-key-with-certs-grouping
- \* local-or-keystore-end-entity-cert-with-key-grouping
- \* keystore-grouping

Each of these groupings are presented in the following subsections.

#### 2.1.3.1. The "encrypted-by-choice-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "encrypted-by-choice-grouping" grouping:

```

| The grouping's name is intended to be parsed "(encrypted-
| by)-(choice)-(grouping)", not as "(encrypted)-(by-
| choice)-(grouping)".

```

```

grouping encrypted-by-choice-grouping
+-- (encrypted-by-choice)
  +--:(symmetric-key-ref)
    | {central-keystore-supported,symmetric-keys}?
    +-- symmetric-key-ref?   ks:symmetric-key-ref
  +--:(asymmetric-key-ref)
    | {central-keystore-supported,asymmetric-keys}?
    +-- asymmetric-key-ref?  ks:asymmetric-key-ref

```

Comments:

- \* This grouping defines a "choice" statement with options to reference either a symmetric or an asymmetric key configured in the keystore.
- \* This grouping is usable only when the keystore module is implemented. Servers defining custom keystore locations MUST augment in alternate "encrypted-by" references to the alternate locations.

#### 2.1.3.2. The "asymmetric-key-certificate-ref-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "asymmetric-key-certificate-ref-grouping" grouping:

```

grouping asymmetric-key-certificate-ref-grouping
+-- asymmetric-key?   ks:asymmetric-key-ref
| {central-keystore-supported,asymmetric-keys}?
+-- certificate?      leafref

```

## Comments:

- \* This grouping defines a reference to a certificate in two parts: the first being the name of the asymmetric key the certificate is associated with, and the second being the name of the certificate itself.
- \* This grouping is usable only when the keystore module is implemented. Servers defining custom keystore locations MAY define an alternate grouping for references to the alternate locations.

## 2.1.3.3. The "local-or-keystore-symmetric-key-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "local-or-keystore-symmetric-key-grouping" grouping:

```
grouping local-or-keystore-symmetric-key-grouping
  +-- (local-or-keystore)
    +--:(local) {local-definitions-supported,symmetric-keys}?
      |   +-- local-definition
      |   +---u ct:symmetric-key-grouping
    +--:(keystore) {central-keystore-supported,symmetric-keys}?
      +-- keystore-reference?   ks:symmetric-key-ref
```

## Comments:

- \* The "local-or-keystore-symmetric-key-grouping" grouping is provided solely as convenience to downstream modules that wish to offer an option for whether a symmetric key is defined locally or as a reference to a symmetric key in the keystore.
- \* A "choice" statement is used to expose the various options. Each option is enabled by a "feature" statement. Additional "case" statements MAY be augmented in if, e.g., there is a need to reference a symmetric key in an alternate location.
- \* For the "local-definition" option, the definition uses the "symmetric-key-grouping" grouping discussed in Section 2.1.4.3 of [I-D.ietf-netconf-crypto-types].
- \* For the "keystore" option, the "keystore-reference" is an instance of the "symmetric-key-ref" discussed in Section 2.1.2.

## 2.1.3.4. The "local-or-keystore-asymmetric-key-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "local-or-keystore-asymmetric-key-grouping" grouping:

```

grouping local-or-keystore-asymmetric-key-grouping
  +-- (local-or-keystore)
    +--:(local) {local-definitions-supported,asymmetric-keys}?
      |   +-- local-definition
      |       +---u ct:asymmetric-key-pair-grouping
    +--:(keystore) {central-keystore-supported,asymmetric-keys}?
      +-- keystore-reference?   ks:asymmetric-key-ref

```

Comments:

- \* The "local-or-keystore-asymmetric-key-grouping" grouping is provided solely as convenience to downstream modules that wish to offer an option for whether an asymmetric key is defined locally or as a reference to an asymmetric key in the keystore.
- \* A "choice" statement is used to expose the various options. Each option is enabled by a "feature" statement. Additional "case" statements MAY be augmented in if, e.g., there is a need to reference an asymmetric key in an alternate location.
- \* For the "local-definition" option, the definition uses the "asymmetric-key-pair-grouping" grouping discussed in Section 2.1.4.5 of [I-D.ietf-netconf-crypto-types].
- \* For the "keystore" option, the "keystore-reference" is an instance of the "asymmetric-key-ref" typedef discussed in Section 2.1.2.

#### 2.1.3.5. The "local-or-keystore-asymmetric-key-with-certs-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "local-or-keystore-asymmetric-key-with-certs-grouping" grouping:

```

grouping local-or-keystore-asymmetric-key-with-certs-grouping
  +-- (local-or-keystore)
    +--:(local) {local-definitions-supported,asymmetric-keys}?
      |   +-- local-definition
      |       +---u ct:asymmetric-key-pair-with-certs-grouping
    +--:(keystore) {central-keystore-supported,asymmetric-keys}?
      +-- keystore-reference?   ks:asymmetric-key-ref

```

Comments:

- \* The "local-or-keystore-asymmetric-key-with-certs-grouping" grouping is provided solely as convenience to downstream modules that wish to offer an option for whether an asymmetric key is defined locally or as a reference to an asymmetric key in the keystore.

- \* A "choice" statement is used to expose the various options. Each option is enabled by a "feature" statement. Additional "case" statements MAY be augmented in if, e.g., there is a need to reference an asymmetric key in an alternate location.
- \* For the "local-definition" option, the definition uses the "asymmetric-key-pair-with-certs-grouping" grouping discussed in Section 2.1.4.11 of [I-D.ietf-netconf-crypto-types].
- \* For the "keystore" option, the "keystore-reference" is an instance of the "asymmetric-key-ref" typedef discussed in Section 2.1.2.

#### 2.1.3.6. The "local-or-keystore-end-entity-cert-with-key-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "local-or-keystore-end-entity-cert-with-key-grouping" grouping:

```

grouping local-or-keystore-end-entity-cert-with-key-grouping
  +-- (local-or-keystore)
    +---:(local) {local-definitions-supported,asymmetric-keys}?
      |   +-- local-definition
      |     +---u ct:asymmetric-key-pair-with-cert-grouping
    +---:(keystore) {central-keystore-supported,asymmetric-keys}?
      +-- keystore-reference
        +---u asymmetric-key-certificate-ref-grouping

```

Comments:

- \* The "local-or-keystore-end-entity-cert-with-key-grouping" grouping is provided solely as convenience to downstream modules that wish to offer an option for whether a symmetric key is defined locally or as a reference to a symmetric key in the keystore.
- \* A "choice" statement is used to expose the various options. Each option is enabled by a "feature" statement. Additional "case" statements MAY be augmented in if, e.g., there is a need to reference a symmetric key in an alternate location.
- \* For the "local-definition" option, the definition uses the "asymmetric-key-pair-with-certs-grouping" grouping discussed in Section 2.1.4.11 of [I-D.ietf-netconf-crypto-types].
- \* For the "keystore" option, the "keystore-reference" uses the "asymmetric-key-certificate-ref-grouping" grouping discussed in Section 2.1.3.2.

### 2.1.3.7. The "keystore-grouping" Grouping

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

```

grouping keystore-grouping
  +-- asymmetric-keys {asymmetric-keys}?
  |   +-- asymmetric-key* [name]
  |   |   +-- name? string
  |   |   +---u ct:asymmetric-key-pair-with-certs-grouping
  +-- symmetric-keys {symmetric-keys}?
  |   +-- symmetric-key* [name]
  |   |   +-- name? string
  |   |   +---u ct:symmetric-key-grouping

```

Comments:

- \* The "keystore-grouping" grouping defines a keystore instance as being composed of symmetric and asymmetric keys. The structure for the symmetric and asymmetric keys is essentially the same, being a "list" inside a "container".
- \* For asymmetric keys, each "asymmetric-key" uses the "asymmetric-key-pair-with-certs-grouping" grouping discussed in Section 2.1.4.11 of [I-D.ietf-netconf-crypto-types].
- \* For symmetric keys, each "symmetric-key" uses the "symmetric-key-grouping" grouping discussed in Section 2.1.4.3 of [I-D.ietf-netconf-crypto-types].

### 2.1.4. Protocol-accessible Nodes

The following tree diagram [RFC8340] lists all the protocol-accessible nodes defined in the "ietf-keystore" module, without expanding the "grouping" statements:

```

module: ietf-keystore
  +--rw keystore
  |   +---u keystore-grouping

```

The following tree diagram [RFC8340] lists all the protocol-accessible nodes defined in the "ietf-keystore" module, with all "grouping" statements expanded, enabling the keystore's full structure to be seen:

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

```

module: ietf-keystore
  +--rw keystore
    +--rw asymmetric-keys {asymmetric-keys}?
      +--rw asymmetric-key* [name]
        +--rw name string
        +--rw public-key-format identityref
        +--rw public-key binary
        +--rw private-key-format? identityref
        +--rw (private-key-type)
          +--:(cleartext-private-key)
            +--rw cleartext-private-key? binary
          +--:(hidden-private-key) {hidden-keys}?
            +--rw hidden-private-key? empty
          +--:(encrypted-private-key) {private-key-encryption}?
            +--rw encrypted-private-key
              +--rw encrypted-by
                +--rw (encrypted-by-choice)
                  +--:(symmetric-key-ref)
                    {central-keystore-supported,symme\
tric-keys}?
                  +--rw symmetric-key-ref?
                    ks:symmetric-key-ref
                  +--:(asymmetric-key-ref)
                    {central-keystore-supported,asymm\
etric-keys}?
                  +--rw asymmetric-key-ref?
                    ks:asymmetric-key-ref
              +--rw encrypted-value-format identityref
              +--rw encrypted-value binary
            +--rw certificates
              +--rw certificate* [name]
                +--rw name string
                +--rw cert-data end-entity-cert-cms
                +--n certificate-expiration
                  {certificate-expiration-notification}?
                +-- expiration-date yang:date-and-time
              +--x generate-certificate-signing-request
                {certificate-signing-request-generation}?
              +--w input
                +--w csr-info ct:csr-info
              +--ro output
                +--ro certificate-signing-request ct:csr
      +--rw symmetric-keys {symmetric-keys}?
        +--rw symmetric-key* [name]
          +--rw name string
          +--rw key-format? identityref

```

```

    +--rw (key-type)
      +--:(cleartext-key)
      |   +--rw cleartext-key?    binary
      +--:(hidden-key) {hidden-keys}?
      |   +--rw hidden-key?       empty
      +--:(encrypted-key) {symmetric-key-encryption}?
      +--rw encrypted-key
      |   +--rw encrypted-by
      |   |   +--rw (encrypted-by-choice)
      |   |   |   +--:(symmetric-key-ref)
      |   |   |   |   {central-keystore-supported,symme\
      |   |   |   |   tric-keys}?
      |   |   |   |   +--rw symmetric-key-ref?
      |   |   |   |   |   ks:symmetric-key-ref
      |   |   |   |   +--:(asymmetric-key-ref)
      |   |   |   |   |   {central-keystore-supported,asymm\
      |   |   |   |   |   etric-keys}?
      |   |   |   |   |   +--rw asymmetric-key-ref?
      |   |   |   |   |   |   ks:asymmetric-key-ref
      |   |   |   |   +--rw encrypted-value-format    identityref
      |   |   |   +--rw encrypted-value                binary

```

#### Comments:

- \* Protocol-accessible nodes are those nodes that are accessible when the module is "implemented", as described in Section 5.6.5 of [RFC7950].
- \* The protocol-accessible nodes for the "ietf-keystore" module are an instance of the "keystore-grouping" grouping discussed in Section 2.1.3.7.
- \* The reason for why "keystore-grouping" exists separate from the protocol-accessible nodes definition is so as to enable instances of the keystore to be instantiated in other locations, as may be needed or desired by some modules.

## 2.2. Example Usage

The examples in this section are encoded using XML, such as might be the case when using the NETCONF protocol. Other encodings MAY be used, such as JSON when using the RESTCONF protocol.

### 2.2.1. A Keystore Instance

The following example illustrates keys in <running>. Please see Section 3 for an example illustrating built-in values in <operational>.



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

```

<keystore
  xmlns="urn:ietf:params:xml:ns:yang:ietf-keystore"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">

  <symmetric-keys>
    <symmetric-key>
      <name>cleartext-symmetric-key</name>
      <key-format>ct:octet-string-key-format</key-format>
      <cleartext-key>BASE64VALUE=</cleartext-key>
    </symmetric-key>
    <symmetric-key>
      <name>hidden-symmetric-key</name>
      <hidden-key/>
    </symmetric-key>
    <symmetric-key>
      <name>encrypted-symmetric-key</name>
      <key-format>ct:one-symmetric-key-format</key-format>
      <encrypted-key>
        <encrypted-by>
          <asymmetric-key-ref>hidden-asymmetric-key</asymmetric-k\
ey-ref>
        </encrypted-by>
        <encrypted-value-format>
          ct:cms-enveloped-data-format
        </encrypted-value-format>
        <encrypted-value>BASE64VALUE=</encrypted-value>
      </encrypted-key>
    </symmetric-key>
  </symmetric-keys>

  <asymmetric-keys>
    <asymmetric-key>
      <name>ssh-rsa-key</name>
      <public-key-format>
        ct:ssh-public-key-format
      </public-key-format>
      <public-key>BASE64VALUE=</public-key>
      <private-key-format>
        ct:rsa-private-key-format
      </private-key-format>
      <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
    </asymmetric-key>
    <asymmetric-key>
      <name>ssh-rsa-key-with-cert</name>
      <public-key-format>
        ct:subject-public-key-info-format

```

```
</public-key-format>
<public-key>BASE64VALUE=</public-key>
<private-key-format>
  ct:rsa-private-key-format
</private-key-format>
<cleartext-private-key>BASE64VALUE=</cleartext-private-key>
<certificates>
  <certificate>
    <name>ex-rsa-cert2</name>
    <cert-data>BASE64VALUE=</cert-data>
  </certificate>
</certificates>
</asymmetric-key>
<asymmetric-key>
  <name>raw-private-key</name>
  <public-key-format>
    ct:subject-public-key-info-format
  </public-key-format>
  <public-key>BASE64VALUE=</public-key>
  <private-key-format>
    ct:rsa-private-key-format
  </private-key-format>
  <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
</asymmetric-key>
<asymmetric-key>
  <name>rsa-asymmetric-key</name>
  <public-key-format>
    ct:subject-public-key-info-format
  </public-key-format>
  <public-key>BASE64VALUE=</public-key>
  <private-key-format>
    ct:rsa-private-key-format
  </private-key-format>
  <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
  <certificates>
    <certificate>
      <name>ex-rsa-cert</name>
      <cert-data>BASE64VALUE=</cert-data>
    </certificate>
  </certificates>
</asymmetric-key>
<asymmetric-key>
  <name>ec-asymmetric-key</name>
  <public-key-format>
    ct:subject-public-key-info-format
  </public-key-format>
  <public-key>BASE64VALUE=</public-key>
  <private-key-format>
```

```

        ct:ec-private-key-format
    </private-key-format>
    <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
    <certificates>
        <certificate>
            <name>ex-ec-cert</name>
            <cert-data>BASE64VALUE=</cert-data>
        </certificate>
    </certificates>
</asymmetric-key>
<asymmetric-key>
    <name>hidden-asymmetric-key</name>
    <public-key-format>
        ct:subject-public-key-info-format
    </public-key-format>
    <public-key>BASE64VALUE=</public-key>
    <hidden-private-key/>
    <certificates>
        <certificate>
            <name>builtin-idevid-cert</name>
            <cert-data>BASE64VALUE=</cert-data>
        </certificate>
        <certificate>
            <name>my-ldevid-cert</name>
            <cert-data>BASE64VALUE=</cert-data>
        </certificate>
    </certificates>
</asymmetric-key>
<asymmetric-key>
    <name>encrypted-asymmetric-key</name>
    <public-key-format>
        ct:subject-public-key-info-format
    </public-key-format>
    <public-key>BASE64VALUE=</public-key>
    <private-key-format>
        ct:one-asymmetric-key-format
    </private-key-format>
    <encrypted-private-key>
        <encrypted-by>
            <symmetric-key-ref>encrypted-symmetric-key</symmetric-k\
ey-ref>
        </encrypted-by>
        <encrypted-value-format>
            ct:cms-encrypted-data-format
        </encrypted-value-format>
        <encrypted-value>BASE64VALUE=</encrypted-value>
    </encrypted-private-key>
</asymmetric-key>

```

```

    </asymmetric-keys>
  </keystore>

```

### 2.2.2. A Certificate Expiration Notification

The following example illustrates a "certificate-expiration" notification for a certificate associated with a key configured in the keystore.

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

```

<notification
  xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2018-05-25T00:01:00Z</eventTime>
  <keystore xmlns="urn:ietf:params:xml:ns:yang:ietf-keystore">
    <asymmetric-keys>
      <asymmetric-key>
        <name>hidden-asymmetric-key</name>
        <certificates>
          <certificate>
            <name>my-ldevid-cert</name>
            <certificate-expiration>
              <expiration-date>2018-08-05T14:18:53-05:00</expiration\
-date>
            </certificate-expiration>
          </certificate>
        </certificates>
      </asymmetric-key>
    </asymmetric-keys>
  </keystore>
</notification>

```

### 2.2.3. The "Local or Keystore" Groupings

This section illustrates the various "local-or-keystore" groupings defined in the "ietf-keystore" module, specifically the "local-or-keystore-symmetric-key-grouping" (Section 2.1.3.3), "local-or-keystore-asymmetric-key-grouping" (Section 2.1.3.4), "local-or-keystore-asymmetric-key-with-certs-grouping" (Section 2.1.3.5), and "local-or-keystore-end-entity-cert-with-key-grouping" (Section 2.1.3.6) groupings.

These examples assume the existence of an example module called "ex-keystore-usage" having the namespace "http://example.com/ns/example-keystore-usage".

The ex-keystore-usage module is first presented using tree diagrams [RFC8340], followed by an instance example illustrating all the "local-or-keystore" groupings in use, followed by the YANG module itself.

The following tree diagram illustrates "ex-keystore-usage" without expanding the "grouping" statements:

```

module: ex-keystore-usage
  +--rw keystore-usage
    +--rw symmetric-key* [name]
      |   +--rw name string
      |   +---u ks:local-or-keystore-symmetric-key-grouping
    +--rw asymmetric-key* [name]
      |   +--rw name string
      |   +---u ks:local-or-keystore-asymmetric-key-grouping
    +--rw asymmetric-key-with-certs* [name]
      |   +--rw name
      |   |   string
      |   +---u ks:local-or-keystore-asymmetric-key-with-certs-grouping
    +--rw end-entity-cert-with-key* [name]
      |   +--rw name
      |   |   string
      |   +---u ks:local-or-keystore-end-entity-cert-with-key-grouping

```

The following tree diagram illustrates the "ex-keystore-usage" module, with all "grouping" statements expanded, enabling the usage's full structure to be seen:

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

```

module: ex-keystore-usage
  +--rw keystore-usage
    +--rw symmetric-key* [name]
      |   +--rw name string
      |   +--rw (local-or-keystore)
      |   |   +--:(local) {local-definitions-supported,symmetric-keys}?
      |   |   |   +--rw local-definition
      |   |   |   |   +--rw key-format? identityref
      |   |   |   |   +--rw (key-type)
      |   |   |   |   |   +--:(cleartext-key)
      |   |   |   |   |   |   +--rw cleartext-key? binary
      |   |   |   |   |   +--:(hidden-key) {hidden-keys}?
      |   |   |   |   |   |   +--rw hidden-key? empty
      |   |   |   |   +--:(encrypted-key) {symmetric-key-encryption}?
      |   |   |   |   |   +--rw encrypted-key
      |   |   |   |   |   |   +--rw encrypted-by
      |   |   |   |   |   +--rw encrypted-value-format identityref

```

```

|         +---rw encrypted-value          binary
+---:(keystore)
|   {central-keystore-supported,symmetric-keys}?
|   +---rw keystore-reference?   ks:symmetric-key-ref
+---rw asymmetric-key* [name]
|   +---rw name                  string
+---rw (local-or-keystore)
|   +---:(local) {local-definitions-supported,asymmetric-keys}?
|   |   +---rw local-definition
|   |   |   +---rw public-key-format          identityref
|   |   |   +---rw public-key                binary
|   |   |   +---rw private-key-format?        identityref
|   |   |   +---rw (private-key-type)
|   |   |   |   +---:(cleartext-private-key)
|   |   |   |   |   +---rw cleartext-private-key?  binary
|   |   |   |   +---:(hidden-private-key) {hidden-keys}?
|   |   |   |   |   +---rw hidden-private-key?    empty
|   |   |   |   +---:(encrypted-private-key)
|   |   |   |   |   {private-key-encryption}?
|   |   |   |   |   +---rw encrypted-private-key
|   |   |   |   |   |   +---rw encrypted-by
|   |   |   |   |   |   +---rw encrypted-value-format  identityref
|   |   |   |   |   |   +---rw encrypted-value        binary
|   |   |   +---:(keystore)
|   |   |   |   {central-keystore-supported,asymmetric-keys}?
|   |   |   |   +---rw keystore-reference?   ks:asymmetric-key-ref
+---rw asymmetric-key-with-certs* [name]
|   +---rw name                  string
+---rw (local-or-keystore)
|   +---:(local) {local-definitions-supported,asymmetric-keys}?
|   |   +---rw local-definition
|   |   |   +---rw public-key-format          identityref
|   |   |   |   +---rw public-key                binary
|   |   |   |   +---rw private-key-format?        identityref
|   |   |   |   |   +---rw (private-key-type)
|   |   |   |   |   |   +---:(cleartext-private-key)
|   |   |   |   |   |   |   +---rw cleartext-private-key?  binary
|   |   |   |   |   |   |   +---:(hidden-private-key) {hidden-keys}?
|   |   |   |   |   |   |   |   +---rw hidden-private-key?    empty
|   |   |   |   |   |   |   +---:(encrypted-private-key)
|   |   |   |   |   |   |   |   {private-key-encryption}?
|   |   |   |   |   |   |   |   +---rw encrypted-private-key
|   |   |   |   |   |   |   |   |   +---rw encrypted-by
|   |   |   |   |   |   |   |   |   +---rw encrypted-value-format  identityref
|   |   |   |   |   |   |   |   |   +---rw encrypted-value        binary
|   |   |   +---rw certificates

```

```

    +---rw certificate* [name]
    |   +---rw name                               string
    |   +---rw cert-data
    |   |       end-entity-cert-cms
    |   +---n certificate-expiration
    |   |       {certificate-expiration-notification}?
    |   |       +--- expiration-date      yang:date-and-time
    +---x generate-certificate-signing-request
    |   {certificate-signing-request-generation}?
    |   +---w input
    |   |   +---w csr-info      ct:csr-info
    +---ro output
    |   +---ro certificate-signing-request      ct:csr
+---:(keystore)
|   {central-keystore-supported, asymmetric-keys}?
+---rw keystore-reference?  ks:asymmetric-key-ref
+---rw end-entity-cert-with-key* [name]
|   +---rw name                               string
+---rw (local-or-keystore)
|   +---:(local) {local-definitions-supported, asymmetric-keys}?
|   |   +---rw local-definition
|   |   |   +---rw public-key-format
|   |   |   |   identityref
|   |   +---rw public-key                               binary
|   |   +---rw private-key-format?
|   |   |   identityref
|   +---rw (private-key-type)
|   |   +---:(cleartext-private-key)
|   |   |   +---rw cleartext-private-key?               binary
|   |   +---:(hidden-private-key) {hidden-keys}?
|   |   |   +---rw hidden-private-key?                   empty
|   |   +---:(encrypted-private-key)
|   |   |   {private-key-encryption}?
|   |   |   +---rw encrypted-private-key
|   |   |   |   +---rw encrypted-by
|   |   |   |   +---rw encrypted-value-format      identityref
|   |   |   |   +---rw encrypted-value              binary
|   +---rw cert-data?
|   |   end-entity-cert-cms
|   +---n certificate-expiration
|   |   {certificate-expiration-notification}?
|   |   +--- expiration-date      yang:date-and-time
|   +---x generate-certificate-signing-request
|   |   {certificate-signing-request-generation}?
|   |   +---w input
|   |   |   +---w csr-info      ct:csr-info
|   +---ro output
|   |   +---ro certificate-signing-request      ct:csr

```

```

    +--:(keystore)
      {central-keystore-supported, asymmetric-keys}?
      +--rw keystore-reference
        +--rw asymmetric-key?   ks:asymmetric-key-ref
        |   {central-keystore-supported, asymmetric-keys}\
    }?
      +--rw certificate?         leafref

```

The following example provides two equivalent instances of each grouping, the first being a reference to a keystore and the second being locally-defined. The instance having a reference to a keystore is consistent with the keystore defined in Section 2.2.1. The two instances are equivalent, as the locally-defined instance example contains the same values defined by the keystore instance referenced by its sibling example.

```

<keystore-usage
  xmlns="http://example.com/ns/example-keystore-usage"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">

  <!-- The following two equivalent examples illustrate the -->
  <!-- "local-or-keystore-symmetric-key-grouping" grouping: -->

  <symmetric-key>
    <name>example 1a</name>
    <keystore-reference>cleartext-symmetric-key</keystore-reference>
  </symmetric-key>

  <symmetric-key>
    <name>example 1b</name>
    <local-definition>
      <key-format>ct:octet-string-key-format</key-format>
      <cleartext-key>BASE64VALUE=</cleartext-key>
    </local-definition>
  </symmetric-key>

  <!-- The following two equivalent examples illustrate the -->
  <!-- "local-or-keystore-asymmetric-key-grouping" grouping: -->

  <asymmetric-key>
    <name>example 2a</name>
    <keystore-reference>rsa-asymmetric-key</keystore-reference>
  </asymmetric-key>

  <asymmetric-key>
    <name>example 2b</name>
    <local-definition>

```



```
<public-key-format>
  ct:subject-public-key-info-format
</public-key-format>
<public-key>BASE64VALUE=</public-key>
<private-key-format>
  ct:rsa-private-key-format
</private-key-format>
<cleartext-private-key>BASE64VALUE=</cleartext-private-key>
</local-definition>
</asymmetric-key>

<!-- the following two equivalent examples illustrate      -->
<!-- "local-or-keystore-asymmetric-key-with-certs-grouping": -->

<asymmetric-key-with-certs>
  <name>example 3a</name>
  <keystore-reference>rsa-asymmetric-key</keystore-reference>
</asymmetric-key-with-certs>

<asymmetric-key-with-certs>
  <name>example 3b</name>
  <local-definition>
    <public-key-format>
      ct:subject-public-key-info-format
    </public-key-format>
    <public-key>BASE64VALUE=</public-key>
    <private-key-format>
      ct:rsa-private-key-format
    </private-key-format>
    <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
    <certificates>
      <certificate>
        <name>a locally-defined cert</name>
        <cert-data>BASE64VALUE=</cert-data>
      </certificate>
    </certificates>
  </local-definition>
</asymmetric-key-with-certs>

<!-- The following two equivalent examples illustrate      -->
<!-- "local-or-keystore-end-entity-cert-with-key-grouping": -->

<end-entity-cert-with-key>
  <name>example 4a</name>
  <keystore-reference>
    <asymmetric-key>rsa-asymmetric-key</asymmetric-key>
```

```
    <certificate>ex-rsa-cert</certificate>
  </keystore-reference>
</end-entity-cert-with-key>

<end-entity-cert-with-key>
  <name>example 4b</name>
  <local-definition>
    <public-key-format>
      ct:subject-public-key-info-format
    </public-key-format>
    <public-key>BASE64VALUE=</public-key>
    <private-key-format>
      ct:rsa-private-key-format
    </private-key-format>
    <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
    <cert-data>BASE64VALUE=</cert-data>
  </local-definition>
</end-entity-cert-with-key>

</keystore-usage>
```

Following is the "ex-keystore-usage" module's YANG definition:

```
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 CCCC: A YANG Data Model for a Keystore";
  }

  organization
    "Example Corporation";

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

  description
    "This module illustrates notable groupings defined in
    the 'ietf-keystore' module.";

  revision 2022-03-07 {
    description
      "Initial version";
    reference

```

```
    "RFC CCCC: A YANG Data Model for a Keystore";
  }

  container keystore-usage {
    description
      "An illustration of the various keystore groupings.";
    list symmetric-key {
      key "name";
      leaf name {
        type string;
        description
          "An arbitrary name for this key.";
      }
      uses ks:local-or-keystore-symmetric-key-grouping;
      description
        "An symmetric key that may be configured locally or be a
        reference to a symmetric key in the keystore.";
    }
    list asymmetric-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 asymmetric-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 asymmetric
      key, that may be configured locally or be a reference
      to another certificate (and its associated asymmetric
      key) in the keystore.";
  }
}
```

### 2.3. YANG Module

This YANG module has normative references to [RFC8341] and [I-D.ietf-netconf-crypto-types].

<CODE BEGINS> file "ietf-keystore@2022-03-07.yang"

```
module ietf-keystore {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-keystore";
  prefix ks;

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

  import ietf-crypto-types {
    prefix ct;
    reference
      "RFC AAAA: YANG Data Types and Groupings for Cryptography";
  }

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

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

  description
```

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

Copyright (c) 2021 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

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

```
revision 2022-03-07 {
  description
    "Initial version";
  reference
    "RFC CCCC: A YANG Data Model for a Keystore";
}

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

feature central-keystore-supported {
  description
    "The 'central-keystore-supported' feature indicates that
    the server supports the keystore (i.e., implements the
    'ietf-keystore' module).";
}

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

```
feature asymmetric-keys {
  description
    "The 'asymmetric-keys' feature indicates that the server
    supports asymmetric keys in keystores.";
}

feature symmetric-keys {
  description
    "The 'symmetric-keys' feature indicates that the server
    supports symmetric keys in keystores.";
}

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

typedef symmetric-key-ref {
  type leafref {
    path "/ks:keystore/ks:symmetric-keys/ks:symmetric-key"
      + "/ks:name";
  }
  description
    "This typedef enables modules to easily define a reference
    to a symmetric key stored in the keystore, when this
    module is implemented.";
}

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, when this
    module is implemented.";
}

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

grouping encrypted-by-choice-grouping {
  description
    "A grouping that defines a 'choice' statement that can be
    augmented into the 'encrypted-by' node, present in the
    'symmetric-key-grouping' and 'asymmetric-key-pair-grouping'
    groupings defined in RFC AAAA, enabling references to keys
```

```
    in the keystore, when this module is implemented.";
choice encrypted-by-choice {
  nacm:default-deny-write;
  mandatory true;
  description
    "A choice amongst other symmetric or asymmetric keys.";
case symmetric-key-ref {
  if-feature "central-keystore-supported";
  if-feature "symmetric-keys";
  leaf symmetric-key-ref {
    type ks:symmetric-key-ref;
    description
      "Identifies the symmetric key used to encrypt the
        associated key.";
  }
}
case asymmetric-key-ref {
  if-feature "central-keystore-supported";
  if-feature "asymmetric-keys";
  leaf asymmetric-key-ref {
    type ks:asymmetric-key-ref;
    description
      "Identifies the asymmetric key whose public key
        encrypted the associated key.";
  }
}
}
}

grouping asymmetric-key-certificate-ref-grouping {
  description
    "This grouping defines a reference to a specific certificate
      associated with an asymmetric key stored in the keystore,
      when this module is implemented.";
  leaf asymmetric-key {
    nacm:default-deny-write;
    if-feature "central-keystore-supported";
    if-feature "asymmetric-keys";
    type ks:asymmetric-key-ref;
    must '../certificate';
    description
      "A reference to an asymmetric key in the keystore.";
  }
  leaf certificate {
    nacm:default-deny-write;
    type leafref {
      path "/ks:keystore/ks:asymmetric-keys/ks:asymmetric-key"
        + "[ks:name = current()../asymmetric-key]/";
    }
  }
}
```

```

        + "ks:certificates/ks:certificate/ks:name";
    }
    must '../asymmetric-key';
    description
        "A reference to a specific certificate of the
        asymmetric key in the keystore.";
    }
}

// local-or-keystore-* groupings

grouping local-or-keystore-symmetric-key-grouping {
    description
        "A grouping that expands to allow the symmetric key to be
        either stored locally, i.e., within the using data model,
        or a reference to a symmetric key stored in the keystore.

        Servers that do not 'implement' this module, and hence
        'central-keystore-supported' is not defined, SHOULD
        augment in custom 'case' statements enabling references
        to the alternate keystore locations.";
    choice local-or-keystore {
        nacm:default-deny-write;
        mandatory true;
        description
            "A choice between an inlined definition and a definition
            that exists in the keystore.";
        case local {
            if-feature "local-definitions-supported";
            if-feature "symmetric-keys";
            container local-definition {
                description
                    "Container to hold the local key definition.";
                uses ct:symmetric-key-grouping;
            }
        }
        case keystore {
            if-feature "central-keystore-supported";
            if-feature "symmetric-keys";
            leaf keystore-reference {
                type ks:symmetric-key-ref;
                description
                    "A reference to an symmetric key that exists in
                    the keystore, when this module is implemented.";
            }
        }
    }
}

```



```
grouping local-or-keystore-asymmetric-key-grouping {
  description
    "A grouping that expands to allow the asymmetric key to be
    either stored locally, i.e., within the using data model,
    or a reference to an asymmetric key stored in the keystore.

    Servers that do not 'implement' this module, and hence
    'central-keystore-supported' is not defined, SHOULD
    augment in custom 'case' statements enabling references
    to the alternate keystore locations.";
  choice local-or-keystore {
    nacm:default-deny-write;
    mandatory true;
    description
      "A choice between an inlined definition and a definition
      that exists in the keystore.";
    case local {
      if-feature "local-definitions-supported";
      if-feature "asymmetric-keys";
      container local-definition {
        description
          "Container to hold the local key definition.";
        uses ct:asymmetric-key-pair-grouping;
      }
    }
    case keystore {
      if-feature "central-keystore-supported";
      if-feature "asymmetric-keys";
      leaf keystore-reference {
        type ks:asymmetric-key-ref;
        description
          "A reference to an asymmetric key that exists in
          the keystore, when this module is implemented. The
          intent is to reference just the asymmetric key
          without any regard for any certificates that may
          be associated with it.";
      }
    }
  }
}

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,
    i.e., within the using data model, or a reference to an
    asymmetric key (and its associated certificates) stored
    in the keystore."
```

```
Servers that do not 'implement' this module, and hence
'central-keystore-supported' is not defined, SHOULD
augment in custom 'case' statements enabling references
to the alternate keystore locations.";
choice local-or-keystore {
  nacm:default-deny-write;
  mandatory true;
  description
    "A choice between an inlined definition and a definition
    that exists in the keystore.";
  case local {
    if-feature "local-definitions-supported";
    if-feature "asymmetric-keys";
    container local-definition {
      description
        "Container to hold the local key definition.";
      uses ct:asymmetric-key-pair-with-certs-grouping;
    }
  }
  case keystore {
    if-feature "central-keystore-supported";
    if-feature "asymmetric-keys";
    leaf keystore-reference {
      type ks:asymmetric-key-ref;
      description
        "A reference to an asymmetric-key (and all of its
        associated certificates) in the keystore, when
        this module is implemented.";
    }
  }
}

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

    Servers that do not 'implement' this module, and hence
    'central-keystore-supported' is not defined, SHOULD
    augment in custom 'case' statements enabling references
    to the alternate keystore locations.";
  choice local-or-keystore {
    nacm:default-deny-write;
    mandatory true;
    description
```

```
    "A choice between an inlined definition and a definition
      that exists in the keystore.";
  case local {
    if-feature "local-definitions-supported";
    if-feature "asymmetric-keys";
    container local-definition {
      description
        "Container to hold the local key definition.";
      uses ct:asymmetric-key-pair-with-cert-grouping;
    }
  }
  case keystore {
    if-feature "central-keystore-supported";
    if-feature "asymmetric-keys";
    container keystore-reference {
      uses asymmetric-key-certificate-ref-grouping;
      description
        "A reference to a specific certificate associated with
          an asymmetric key stored in the keystore, when this
          module is implemented.";
    }
  }
}

grouping keystore-grouping {
  description
    "Grouping definition enables use in other contexts. If ever
      done, implementations MUST augment new 'case' statements
      into the various local-or-keystore 'choice' statements to
      supply leafrefs to the model-specific location(s).";
  container asymmetric-keys {
    nacm:default-deny-write;
    if-feature "asymmetric-keys";
    description
      "A list of asymmetric keys.";
    list asymmetric-key {
      key "name";
      description
        "An asymmetric key.";
      leaf name {
        type string;
        description
          "An arbitrary name for the asymmetric key.";
      }
      uses ct:asymmetric-key-pair-with-certs-grouping;
    }
  }
}
```

```
    container symmetric-keys {
      nacm:default-deny-write;
      if-feature "symmetric-keys";
      description
        "A list of symmetric keys.";
      list symmetric-key {
        key "name";
        description
          "A symmetric key.";
        leaf name {
          type string;
          description
            "An arbitrary name for the symmetric key.";
        }
        uses ct:symmetric-key-grouping;
      }
    }
  }

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

  container keystore {
    description
      "A central keystore containing a list of symmetric keys and
      a list of asymmetric keys.";
    nacm:default-deny-write;
    uses keystore-grouping {
      augment "symmetric-keys/symmetric-key/key-type/encrypted-key/"
        + "encrypted-key/encrypted-by" {
        description
          "Augments in a choice statement enabling the encrypting
          key to be any other symmetric or asymmetric key in the
          central keystore.";
        uses encrypted-by-choice-grouping;
      }
      augment "asymmetric-keys/asymmetric-key/private-key-type/"
        + "encrypted-private-key/encrypted-private-key/"
        + "encrypted-by" {
        description
          "Augments in a choice statement enabling the encrypting
          key to be any other symmetric or asymmetric key in the
          central keystore.";
        uses encrypted-by-choice-grouping;
      }
    }
  }
}
```

```

}

<CODE ENDS>

```

### 3. Support for Built-in Keys

In some implementations, a server may support built-in keys. Built-in keys MAY be set during the manufacturing process or be dynamically generated the first time the server is booted or a particular service (e.g., SSH) is enabled.

The primary characteristic of the built-in keys is that they are provided by the system, as opposed to configuration. As such, they are present in <operational> (and <system> [I-D.ma-netmod-with-system], if used). The example below illustrates what the keystore in <operational> might look like for a server in its factory default state.

```

<keystore xmlns="urn:ietf:params:xml:ns:yang:ietf-keystore"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types"
  xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
  or:origin="or:intended">
  <asymmetric-keys>
    <asymmetric-key or:origin="or:system">
      <name>Manufacturer-Generated Hidden Key</name>
      <public-key-format>
        ct:subject-public-key-info-format
      </public-key-format>
      <public-key>BASE64VALUE=</public-key>
      <hidden-private-key/>
      <certificates>
        <certificate>
          <name>Manufacturer-Generated IDevID Cert</name>
          <cert-data>BASE64VALUE=</cert-data>
        </certificate>
      </certificates>
    </asymmetric-key>
  </asymmetric-keys>
</keystore>

```

In order for the built-in keys (and their associated built-in certificates) to be referenced by configuration, the referenced keys and associated certificates MUST first be copied into <running>.

Built-in keys that are "hidden" MUST be copied into <running> using the same key values, so that the server can bind them to the built-in entries.

Built-in keys that are "encrypted" MAY be copied into other parts of the configuration so long as they are otherwise unmodified (e.g., the "encrypted-by" reference cannot be altered).

Built-in keys that are "cleartext" MAY be copied into other parts of the configuration but, by doing so, they lose their association to the built-in entries and any assurances afforded by knowing they are/were built-in.

The built-in keys and built-in associated certificates are immutable by configuration operations. With exception to additional/custom certificates associated to a built-in key, servers MUST ignore attempts to modify any aspect of built-in keys and/or built-in associated certificates.

The following example illustrates how a single built-in key definition from the previous example has been propagated to <running>:

```
<keystore xmlns="urn:ietf:params:xml:ns:yang:ietf-keystore"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
  <asymmetric-keys>
    <asymmetric-key>
      <name>Manufacturer-Generated Hidden Key</name>
      <public-key-format>
        ct:subject-public-key-info-format
      </public-key-format>
      <public-key>BASE64VALUE=</public-key>
      <hidden-private-key/>
      <certificates>
        <certificate>
          <name>Manufacturer-Generated IDevID Cert</name>
          <cert-data>BASE64VALUE=</cert-data>
        </certificate>
        <certificate>
          <name>Deployment-Specific LDevID Cert</name>
          <cert-data>BASE64VALUE=</cert-data>
        </certificate>
      </certificates>
    </asymmetric-key>
  </asymmetric-keys>
</keystore>
```

After the above configuration is applied, <operational> should appear as follows:

```

<keystore xmlns="urn:ietf:params:xml:ns:yang:ietf-keystore"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types"
  xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
  or:origin="or:intended">
  <asymmetric-keys>
    <asymmetric-key or:origin="or:system">
      <name>Manufacturer-Generated Hidden Key</name>
      <public-key-format>
        ct:subject-public-key-info-format
      </public-key-format>
      <public-key>BASE64VALUE=</public-key>
      <hidden-private-key/>
      <certificates>
        <certificate>
          <name>Manufacturer-Generated IDevID Cert</name>
          <cert-data>BASE64VALUE=</cert-data>
        </certificate>
        <certificate or:origin="or:intended">
          <name>Deployment-Specific LDevID Cert</name>
          <cert-data>BASE64VALUE=</cert-data>
        </certificate>
      </certificates>
    </asymmetric-key>
  </asymmetric-keys>
</keystore>

```

#### 4. Encrypting Keys in Configuration

This section describes an approach that enables both the symmetric and asymmetric keys on a server to be encrypted, such that traditional backup/restore procedures can be used without concern for the keys being compromised when in transit.

##### 4.1. Key Encryption Key

The ability to encrypt configured keys is predicated on the existence of a "key encryption key" (KEK). There may be any number of KEKs in a system. A KEK, by its namesake, is a key that is used to encrypt other keys. A KEK MAY be either a symmetric key or an asymmetric key.

If a KEK is a symmetric key, then the server MUST provide an API for administrators to encrypt other keys without needing to know the symmetric key's value. If the KEK is an asymmetric key, then the server MAY provide an API enabling the encryption of other keys or, alternatively, let the administrators do so themselves using the asymmetric key's public half.

A server MUST possess (or be able to possess, in case the KEK has been encrypted by another KEK) a KEK's cleartext value so that it can decrypt the other keys in the configuration at runtime.

#### 4.2. Configuring Encrypted Keys

Each time a new key is configured, it SHOULD be encrypted by a KEK.

In "ietf-crypto-types" [I-D.ietf-netconf-crypto-types], the format for encrypted values is described by identity statements derived from the "symmetrically-encrypted-value-format" and "symmetrically-encrypted-value-format" identity statements.

Implementations SHOULD provide an API that simultaneously generates and encrypts a key (symmetric or asymmetric) using a KEK. Thusly newly generated key cleartext values may never be known to the administrators generating the keys.

In case the server implementation does not provide such an API, then the generating and encrypting steps MAY be performed outside the server, e.g., by an administrator with special access control rights (e.g., an organization's crypto officer).

In either case, the encrypted key can be configured into the keystore using either the "encrypted-key" (for symmetric keys) or the "encrypted-private-key" (for asymmetric keys) nodes. These two nodes contain both the encrypted value as well as a reference to the KEK that encrypted the key.

#### 4.3. Migrating Configuration to Another Server

When a KEK is used to encrypt other keys, migrating the configuration to another server is only possible if the second server has the same KEK. How the second server comes to have the same KEK is discussed in this section.

In some deployments, mechanisms outside the scope of this document may be used to migrate a KEK from one server to another. That said, beware that the ability to do so typically entails having access to the first server but, in many scenarios, the first server may no longer be operational.



In other deployments, an organization's crypto officer, possessing a KEK's cleartext value, configures the same KEK on the second server, presumably as a hidden key or a key protected by access-control (e.g., NACM's "default-deny-all"), so that the cleartext value is not disclosed to regular administrators. However, this approach creates high-coupling to and dependency on the crypto officers that does not scale in production environments.

In order to decouple the crypto officers from the regular administrators, a special KEK, called the "master key" (MK), may be used.

A MK is commonly a globally-unique built-in (see Section 3) asymmetric key. The private key, due to its long lifetime, is hidden (i.e., "hidden-private-key" in Section 2.1.4.5. of [I-D.ietf-netconf-crypto-types]). The public key is often contained in an identity certificate (e.g., IDevID). How to configure a MK during the manufacturing process is outside the scope of this document.

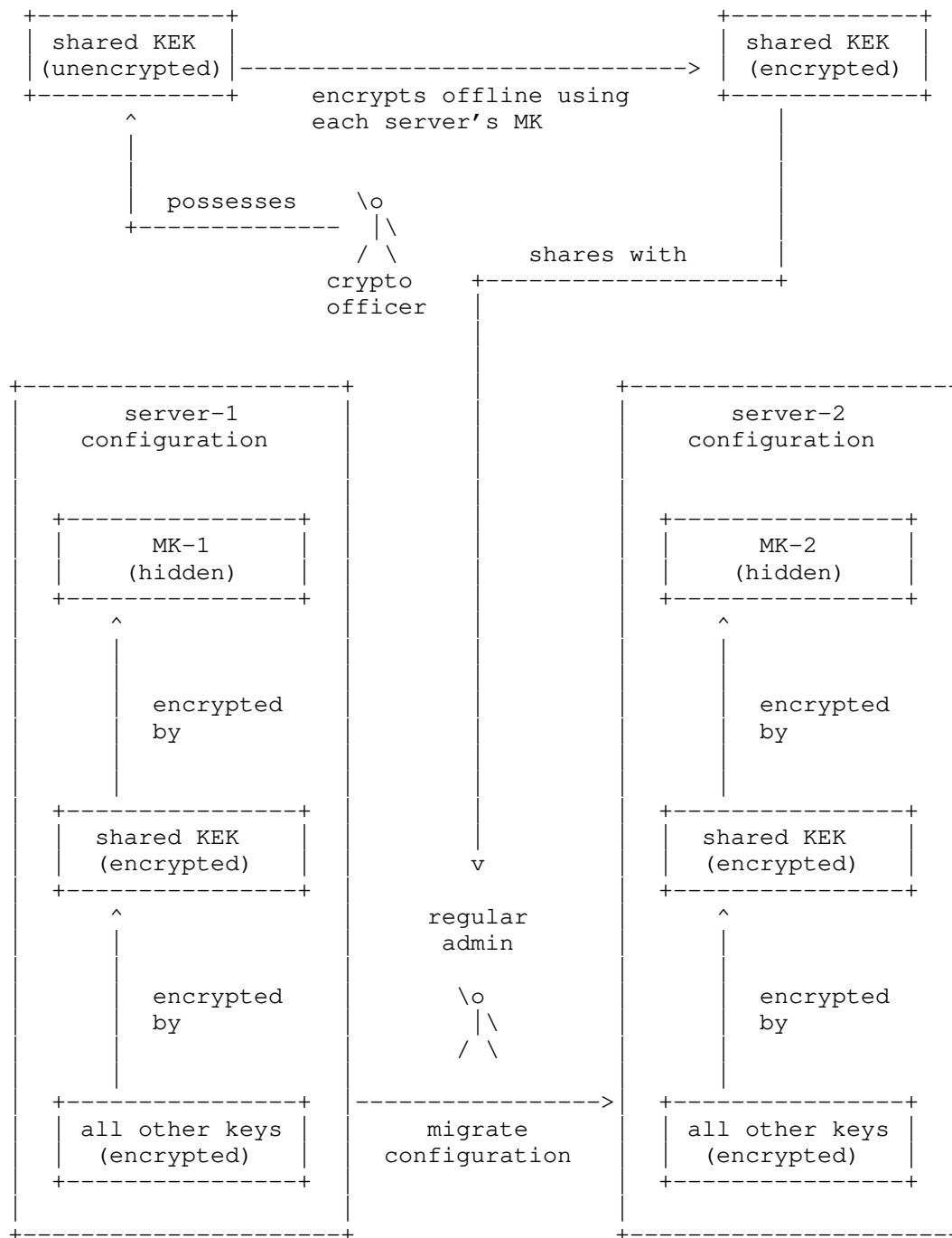
It is RECOMMENDED that MKs are built-in and hidden but, if this is not possible, access control mechanisms like NACM SHOULD be used to limit access to the MK's secret data only to the most trusted authorized clients (e.g., an organization's crypto officer). In this case, it is RECOMMENDED that the MK is not built-in and hence is, effectively, just like a KEK.

Assuming the server has a MK, the MK can be used to encrypt a "shared KEK", which is then used to encrypt the keys configured by regular administrators.

With this extra level of indirection, it is possible for a crypto officer to encrypt the same KEK for a multiplicity of servers offline using the public key contained in their identity certificates. The crypto officer can then safely handoff the encrypted KEKs to the regular administrators responsible for server installations, including migrations.

In order to migrate the configuration from a first server, an administrator would need to make just a single modification to the configuration before loading it onto a second server, which is to replace the encrypted KEK keystore entry from the first server with the encrypted KEK for the second server. Upon doing this, the configuration (containing many encrypted keys) can be loaded into the second server while enabling the second server to decrypt all the encrypted keys in the configuration.

The following diagram illustrates this idea:



## 5. Security Considerations

### 5.1. Security of Data at Rest

The YANG module defined in this document defines a mechanism called a "keystore" that, by its name, suggests that it will protect its contents from unauthorized disclosure and modification.

Security controls for the API (i.e., data in motion) are discussed in Section 5.3, but controls for the data at rest cannot be specified by the YANG module.

In order to satisfy the expectations of a "keystore", it is RECOMMENDED that implementations ensure that the keystore contents are encrypted when persisted to non-volatile memory.

### 5.2. Unconstrained Private Key Usage

This module enables the configuration of private keys without constraints on their usage, e.g., what operations the key is allowed to be used for (e.g., signature, decryption, both).

This module also does not constrain the usage of the associated public keys, other than in the context of a configured certificate (e.g., an identity certificate), in which case the key usage is constrained by the certificate.

### 5.3. The "ietf-keystore" YANG Module

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.

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

Please be aware that this module uses the "cleartext-key" and "cleartext-private-key" nodes from the "ietf-crypto-types" module [I-D.ietf-netconf-crypto-types], where said nodes have the NACM extension "default-deny-all" set, thus preventing uncontrolled read-access to the cleartext key values.

All the writable data nodes defined by this module, both in the "grouping" statements as well as the protocol-accessible "keystore" instance, may be considered sensitive or vulnerable in some network environments.. For instance, any modification to a key or reference to a key may dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been set for all data nodes defined in this module.

This module does not define any "rpc" or "action" statements, and thus the security considerations for such is not provided here.

## 6. IANA Considerations

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

### 6.2. The "YANG Module Names" Registry

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

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

## 7. References

### 7.1. Normative References

- [I-D.ietf-netconf-crypto-types]  
Watsen, K., "YANG Data Types and Groupings for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-21, 14 September 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-crypto-types-21>>.
- [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>>.

## 7.2. Informative References

- [I-D.ietf-netconf-http-client-server]  
Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-08, 14 December 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-http-client-server-08>>.
- [I-D.ietf-netconf-keystore]  
Watsen, K., "A YANG Data Model for a Keystore", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-23, 14 December 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-keystore-23>>.
- [I-D.ietf-netconf-netconf-client-server]  
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-24, 14 December 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-netconf-client-server-24>>.

- [I-D.ietf-netconf-restconf-client-server]  
Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-24, 14 December 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-restconf-client-server-24>>.
- [I-D.ietf-netconf-ssh-client-server]  
Watsen, K., "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-ssh-client-server-26, 14 December 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-ssh-client-server-26>>.
- [I-D.ietf-netconf-tcp-client-server]  
Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tcp-client-server-11, 14 December 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tcp-client-server-11>>.
- [I-D.ietf-netconf-tls-client-server]  
Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-26, 14 December 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tls-client-server-26>>.
- [I-D.ietf-netconf-trust-anchors]  
Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-16, 14 December 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-trust-anchors-16>>.
- [I-D.ma-netmod-with-system]  
Ma, Q., Watsen, K., Wu, Q., Chong, F., and J. Lindblad, "System-defined Configuration", Work in Progress, Internet-Draft, draft-ma-netmod-with-system-02, 14 February 2022, <<https://datatracker.ietf.org/doc/html/draft-ma-netmod-with-system-02>>.
- [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-2018]  
IEEE SA-Standards Board, "IEEE Standard for Local and metropolitan area networks - Secure Device Identity", August 2018, <[https://standards.ieee.org/standard/802\\_1AR-2018.html](https://standards.ieee.org/standard/802_1AR-2018.html)>.

## Appendix A. Change Log

This section is to be removed before publishing as an RFC.

### A.1. 00 to 01

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

### A.2. 01 to 02

- \* Added back 'generate-private-key' action.
- \* Removed 'RESTRICTED' enum from the 'private-key' leaf type.

- \* Fixed up a few description statements.

#### A.3. 02 to 03

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

#### A.4. 03 to 04

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

#### A.5. 04 to 05

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

#### A.6. 05 to 06

- \* Added feature "local-keys-supported"
- \* Added nacm:default-deny-all and nacm:default-deny-write
- \* Renamed generate-asymmetric-key to generate-hidden-key
- \* Added an install-hidden-key action



- \* Moved actions inside fo the "asymmetric-key" container
- \* Moved some groupings to draft-ietf-netconf-crypto-types

#### A.7. 06 to 07

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

#### A.8. 07 to 08

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

#### A.9. 08 to 09

- \* Added a 'description' statement to the 'must' in the /keystore/asymmetric-key node explaining that the descendant values may exist in <operational> only, and that implementation MUST assert that the values are either configured or that they exist in <operational>.
- \* Copied above 'must' statement (and description) into the local-or-keystore-asymmetric-key-grouping, local-or-keystore-asymmetric-key-with-certs-grouping, and local-or-keystore-end-entity-cert-with-key-grouping statements.

#### A.10. 09 to 10

- \* Updated draft title to match new truststore draft title
- \* Moved everything under a top-level 'grouping' to enable use in other contexts.
- \* Renamed feature from 'local-keys-supported' to 'local-definitions-supported' (same name used in truststore)
- \* Removed the either-all-or-none 'must' expressions for the key's 3-tuple values (since the values are now 'mandatory true' in crypto-types)

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

## A.11. 10 to 11

- \* Replaced typedef asymmetric-key-certificate-ref with grouping asymmetric-key-certificate-ref-grouping.
- \* Added feature feature 'key-generation'.
- \* Cloned groupings symmetric-key-grouping, asymmetric-key-pair-grouping, asymmetric-key-pair-with-cert-grouping, and asymmetric-key-pair-with-certs-grouping from crypto-keys, augmenting into each new case statements for values that have been encrypted by other keys in the keystore. Refactored keystore model to use these groupings.
- \* Added new 'symmetric-keys' lists, as a sibling to the existing 'asymmetric-keys' list.
- \* Added RPCs (not actions) 'generate-symmetric-key' and 'generate-asymmetric-key' to \*return\* a (potentially encrypted) key.

## A.12. 11 to 12

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

## A.13. 12 to 13

- \* Updated examples to incorporate new "key-format" identities.
- \* Made the two "generate-\*-key" RPCs be "action" statements instead.

## A.14. 13 to 14

- \* Updated YANG module and examples to incorporate the new iana-\*-algorithm modules in the crypto-types draft..

## A.15. 14 to 15

- \* Added new "Support for Built-in Keys" section.

- \* Added 'must' expressions asserting that the 'key-format' leaf whenever an encrypted key is specified.
- \* Added local-or-keystore-symmetric-key-grouping for PSK support.

## A.16. 15 to 16

- \* Moved the generate key actions to ietf-crypt-types as RPCs, which are augmented by ietf-keystore to support encrypted keys. Examples updated accordingly.
- \* Added a SSH certificate-based key (RFC 6187) and a raw private key to the example instance document (partly so they could be referenced by examples in the SSH and TLS client/server drafts.

## A.17. 16 to 17

- \* Removed augments to the "generate-symmetric-key" and "generate-asymmetric-key" groupings.
- \* Removed "generate-symmetric-key" and "generate-asymmetric-key" examples.
- \* Removed the "algorithm" nodes from remaining examples.
- \* Updated the "Support for Built-in Keys" section.
- \* Added new section "Encrypting Keys in Configuration".
- \* Added a "Note to Reviewers" note to first page.

## A.18. 17 to 18

- \* Removed dangling/unnecessary ref to RFC 8342.
- \* r/MUST/SHOULD/ wrt strength of keys being configured over transports.
- \* Added an example for the "certificate-expiration" notification.
- \* Clarified that OS MAY have a multiplicity of underlying keystores and/or HSMs.
- \* Clarified expected behavior for "built-in" keys in <operational>
- \* Clarified the "Migrating Configuration to Another Server" section.

- \* Expanded "Data Model Overview section(s) [remove "wall" of tree diagrams].
- \* Updated the Security Considerations section.

## A.19. 18 to 19

- \* Updated examples to reflect new "cleartext-" prefix in the cryptotypes draft.

## A.20. 19 to 20

- \* Addressed SecDir comments from Magnus Nystroem and Sandra Murphy.

## A.21. 20 to 21

- \* Added a "Unconstrained Private Key Usage" Security Consideration to address concern raised by SecDir.
- \* (Editorial) Removed the output of "grouping" statements in the tree diagrams for the "ietf-keystore" and "ex-keystore-usage" modules.
- \* Addressed comments raised by YANG Doctor.

## A.22. 21 to 22

- \* Added prefixes to 'path' statements per trust-anchors/issues/1
- \* Renamed feature "keystore-supported" to "central-keystore-supported".
- \* Associated with above, generally moved text to refer to a "central" keystore.
- \* Aligned modules with 'pyang -f' formatting.
- \* Fixed nits found by YANG Doctor reviews.

## A.23. 22 to 23

- \* Updated 802.1AR ref to latest version
- \* Replaced "base64encodedvalue==" with "BASE64VALUE=" in examples.
- \* Minor editorial nits

## A.24. 23 to 24

- \* Added features "asymmetric-keys" and "symmetric-keys"
- \* fixup the 'WG Web' and 'WG List' lines in YANG module(s)
- \* fixup copyright (i.e., s/Simplified/Revised/) in YANG module(s)
- \* Added Informative reference to ma-netmod-with-system

## Acknowledgements

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

## Author's Address

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

NETCONF Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: September 14, 2017

K. Watsen  
Juniper Networks  
G. Wu  
Cisco Networks  
J. Schoenwaelder  
Jacobs University Bremen  
March 13, 2017

NETCONF Client and Server Models  
draft-ietf-netconf-netconf-client-server-02

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 "2017-03-13" --> the publication date of this draft

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

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

#### 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 <http://datatracker.ietf.org/drafts/current/>.

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

This Internet-Draft will expire on September 14, 2017.

#### Copyright Notice

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

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

## Table of Contents

1. Introduction . . . . .	3
1.1. Terminology . . . . .	4
1.2. Tree Diagrams . . . . .	4
2. The NETCONF Client Model . . . . .	4
2.1. Tree Diagram . . . . .	5
2.2. Example Usage . . . . .	8
2.3. YANG Model . . . . .	10
3. The NETCONF Server Model . . . . .	19
3.1. Tree Diagram . . . . .	20
3.2. Example Usage . . . . .	23
3.3. YANG Model . . . . .	26
4. Design Considerations . . . . .	37
4.1. Support all NETCONF transports . . . . .	37
4.2. Enable each transport to select which keys to use . . . . .	37
4.3. Support authenticating NETCONF clients certificates . . . . .	37
4.4. Support mapping authenticated NETCONF client certificates to usernames . . . . .	38
4.5. Support both listening for connections and call home . . . . .	38
4.6. For Call Home connections . . . . .	38
4.6.1. Support more than one NETCONF client . . . . .	38
4.6.2. Support NETCONF clients having more than one endpoint . . . . .	38
4.6.3. Support a reconnection strategy . . . . .	38
4.6.4. Support both persistent and periodic connections . . . . .	39
4.6.5. Reconnection strategy for periodic connections . . . . .	39
4.6.6. Keep-alives for persistent connections . . . . .	39
4.6.7. Customizations for periodic connections . . . . .	39
5. Security Considerations . . . . .	39
6. IANA Considerations . . . . .	41
6.1. The IETF XML Registry . . . . .	41
6.2. The YANG Module Names Registry . . . . .	41
7. Acknowledgements . . . . .	41
8. References . . . . .	42
8.1. Normative References . . . . .	42
8.2. Informative References . . . . .	43
Appendix A. Change Log . . . . .	44
A.1. server-model-09 to 00 . . . . .	44
A.2. 00 to 01 . . . . .	44
A.3. 01 to 02 . . . . .	44
Appendix B. Open Issues . . . . .	44
Authors' Addresses . . . . .	44

## 1. Introduction

This document defines two YANG [RFC7950] 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.

NETCONF is defined by [RFC6241]. SSH is defined by [RFC4252], [RFC4253], and [RFC4254]. TLS is defined by [RFC5246]. NETCONF Call Home is defined by [RFC8071]).

### 1.1. Terminology

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

### 1.2. Tree Diagrams

A simplified graphical representation of the data models is used in this document. The meaning of the symbols in these diagrams is as follows:

- o Brackets "[" and "]" enclose list keys.
- o Braces "{" and "}" enclose feature names, and indicate that the named feature must be present for the subtree to be present.
- o Abbreviations before data node names: "rw" means configuration (read-write) and "ro" state data (read-only).
- o Symbols after data node names: "?" means an optional node, "!" means a presence container, and "\*" denotes a list and leaf-list.
- o Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- o Ellipsis ("...") stands for contents of subtrees that are not shown.

## 2. 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.

## 2.1. Tree Diagram

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

```

module: ietf-netconf-client
  +--rw netconf-client
    +--rw initiate {initiate}?
      +--rw netconf-server* [name]
        +--rw name string
        +--rw (transport)
          +--:(ssh) {ssh-initiate}?
            +--rw ssh
              +--rw endpoints
                +--rw endpoint* [name]
                  +--rw name string
                  +--rw address inet:host
                  +--rw port? inet:port-number
              +--rw server-auth
                +--rw trusted-ssh-host-keys?
                  | -> /ks:keystore/trusted-host-keys/name
                +--rw trusted-ca-certs? leafref
                  | {sshcom:ssh-x509-certs}?
                +--rw trusted-server-certs? leafref
                  | {sshcom:ssh-x509-certs}?
              +--rw client-auth
                +--rw username? string
                +--rw (auth-type)?
                  +--:(certificate)
                    | +--rw certificate? leafref
                    | | {sshcom:ssh-x509-certs}?
                  +--:(public-key)
                    | +--rw public-key?
                    | | -> /ks:keystore/keys/key/name
                  +--:(password)
                    +--rw password? union
              +--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
|         |  +--rw compression
|         |  |  +--rw compression-alg*  identityref
|         |  +---:(tls) {tls-initiate}?
|         |  +--rw tls
|         |  |  +--rw endpoints
|         |  |  |  +--rw endpoint* [name]
|         |  |  |  |  +--rw name      string
|         |  |  |  |  +--rw address   inet:host
|         |  |  |  |  +--rw port?    inet:port-number
|         |  |  +--rw server-auth
|         |  |  |  +--rw trusted-ca-certs?  leafref
|         |  |  |  +--rw trusted-server-certs?  leafref
|         |  |  +--rw client-auth
|         |  |  |  +--rw (auth-type)?
|         |  |  |  |  +---:(certificate)
|         |  |  |  |  |  +--rw certificate?  leafref
|         |  |  +--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 idle-timeout?  uint32
|         |  |  |  |  |  +--rw keep-alives
|         |  |  |  |  |  |  +--rw max-wait?    uint16
|         |  |  |  |  |  |  +--rw max-attempts?  uint8
|         |  |  |  +---:(periodic-connection)
|         |  |  |  |  +--rw periodic!
|         |  |  |  |  |  +--rw idle-timeout?    uint16
|         |  |  |  |  |  +--rw reconnect-timeout?  uint16
|         |  +--rw reconnect-strategy
|         |  |  +--rw start-with?  enumeration
|         |  |  +--rw max-attempts?  uint8
+--rw listen {listen}?
+--rw max-sessions?  uint16
+--rw idle-timeout?  uint16
+--rw endpoint* [name]
+--rw name  string
+--rw (transport)
+--rw (ssh) {ssh-listen}?
|  +--rw ssh

```

```

+--rw address?                inet:ip-address
+--rw port?                   inet:port-number
+--rw server-auth
|   +--rw trusted-ssh-host-keys?
|   |   -> /ks:keystore/trusted-host-keys/name
|   +--rw trusted-ca-certs?    leafref
|   |   {sshcom:ssh-x509-certs}?
|   +--rw trusted-server-certs? leafref
|   |   {sshcom:ssh-x509-certs}?
+--rw client-auth
|   +--rw username?            string
|   +--rw (auth-type)?
|   |   +--:(certificate)
|   |   |   +--rw certificate?    leafref
|   |   |   |   {sshcom:ssh-x509-certs}?
|   |   +--:(public-key)
|   |   |   +--rw public-key?
|   |   |   |   -> /ks:keystore/keys/key/name
|   |   +--:(password)
|   |   |   +--rw password?        union
+--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
+--rw compression
|   +--rw compression-alg*     identityref
+--:(tls) {tls-listen}?
+--rw tls
|   +--rw address?            inet:ip-address
|   +--rw port?              inet:port-number
|   +--rw server-auth
|   |   +--rw trusted-ca-certs?    leafref
|   |   +--rw trusted-server-certs? leafref
|   +--rw client-auth
|   |   +--rw (auth-type)?
|   |   |   +--:(certificate)
|   |   |   |   +--rw certificate?    leafref
+--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 NETCONF client to initiate connections, using both the SSH and TLS transport protocols, as well as listening for call-home connections, again using both the SSH and TLS transport protocols.

This example is consistent with the examples presented in Section 2.2 of [I-D.ietf-netconf-keystore].

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

  <!-- NETCONF servers to initiate connections to -->
  <initiate>
    <netconf-server>
      <name>corp-fw1</name>
      <ssh>
        <endpoints>
          <endpoint>
            <name>corp-fw1.example.com</name>
            <address>corp-fw1.example.com</address>
          </endpoint>
          <endpoint>
            <name>corp-fw2.example.com</name>
            <address>corp-fw2.example.com</address>
          </endpoint>
        </endpoints>
        <server-auth>
          <trusted-server-certs>deployment-specific-ca-certs</trusted-server-cer
ts>
        </server-auth>
        <client-auth>
          <username>foobar</username>
          <public-key>ex-rsa-key</public-key>
        </client-auth>
      </ssh>
    </netconf-server>
  </initiate>

  <!-- endpoints to listen for NETCONF Call Home connections on -->
  <listen>
    <endpoint>
      <name>Intranet-facing listener</name>
      <ssh>
        <address>11.22.33.44</address>
        <server-auth>
          <trusted-ca-certs>deployment-specific-ca-certs</trusted-ca-certs>
          <trusted-server-certs>explicitly-trusted-server-certs</trusted-server-
certs>
          <trusted-ssh-host-keys>explicitly-trusted-ssh-host-keys</trusted-ssh-h
ost-keys>
        </server-auth>
        <client-auth>
          <username>foobar</username>
          <public-key>ex-rsa-key</public-key>
        </client-auth>
      </ssh>
    </endpoint>
  </listen>
</netconf-client>
```

### 2.3. YANG Model

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

```
<CODE BEGINS> file "ietf-netconf-client@2017-03-13.yang"

module ietf-netconf-client {
  yang-version 1.1;

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

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

  import ietf-ssh-client {
    prefix ss;
    revision-date 2017-03-13; // stable grouping definitions
    reference
      "RFC YYYY: SSH Client and Server Models";
  }

  import ietf-tls-client {
    prefix ts;
    revision-date 2017-03-13; // stable grouping definitions
    reference
      "RFC ZZZZ: TLS Client and Server Models";
  }

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

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

    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) 2014 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

```
revision "2017-03-13" {
  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 {
  description
    "Top-level container for NETCONF client configuration.";

  container initiate {
    if-feature initiate;
    description
      "Configures client initiating underlying TCP connections.";
    list netconf-server {
      key name;
      description
        "List of NETCONF servers the NETCONF client is to initiate
        connections to.";
      leaf name {
        type string;
        description
          "An arbitrary name for the NETCONF server.";
      }
      choice transport {
        mandatory true;
      }
    }
  }
}
```

```
description
  "Selects between available transports.";

case ssh {
  if-feature ssh-initiate;
  container ssh {
    description
      "Specifies SSH-specific transport configuration.";
    uses endpoints-container {
      refine endpoints/endpoint/port {
        default 830;
      }
    }
    uses ss:ssh-client-grouping;
  }
} // end ssh

case tls {
  if-feature tls-initiate;
  container tls {
    description
      "Specifies TLS-specific transport configuration.";
    uses endpoints-container {
      refine endpoints/endpoint/port {
        default 6513;
      }
    }
    uses ts:tls-client-grouping;
  }
} // end tls

} // end transport

container connection-type {
  description
    "Indicates the kind of connection to use.";
  choice connection-type {
    description
      "Selects between available connection types.";
    case persistent-connection {
      container persistent {
        presence true;
        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.";

```
leaf idle-timeout {
  type uint32;
  units "seconds";
  default 86400; // one day;
  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 client will never drop
     a session because it is idle. Sessions that
     have a notification subscription active are
     never dropped.";
}
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.";
  reference
    "RFC 8071: NETCONF Call Home and RESTCONF Call
     Home, Section 3.1, item S6";
  leaf max-wait {
    type uint16 {
      range "1..max";
    }
    units seconds;
    default 30;
    description
      "Sets the amount of time in seconds after which
       if no data has been received from the 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 true;
        description
            "Periodically connect to the NETCONF server, so that
             the NETCONF server may deliver messages pending for
             the NETCONF client. The NETCONF server must close
             the connection when it is ready to release it. Once
             the connection has been closed, the NETCONF client
             will restart its timer until the next connection.";
        leaf idle-timeout {
            type uint16;
            units "seconds";
            default 300; // five minutes
            description
                "Specifies the maximum number of seconds that a
                 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.";
        }
        leaf reconnect-timeout {
            type uint16 {
                range "1..max";
            }
            units minutes;
            default 60;
            description
                "Sets the maximum amount of unconnected time the
                 NETCONF client will wait before re-establishing
                 a connection to the NETCONF server. The NETCONF
                 client may initiate a connection before this
                 time if desired (e.g., to set configuration).";
        }
    }
}
}
}
}
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.";
    }
  }
  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;
  description
    "Configures client accepting call-home TCP connections.";

  leaf max-sessions {
    type uint16;
```

```
    default 0;
    description
        "Specifies the maximum number of concurrent sessions
        that can be active at one time. The value 0 indicates
        that no artificial session limit should be used.";
}

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;
    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 for call-home connections.";
                }
                leaf port {
                    type inet:port-number;
                    default 4334;
                    description
```

```
        "The port number to listen for call-home connections.";
    }
    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 for call-home connections.";
        }
        leaf port {
            type inet:port-number;
            default 4335;
            description
                "The port number to listen for call-home connections.";
        }
        uses ts:tls-client-grouping;
    }
}
} // end transport
} // end endpoint
} // end listen

} // end netconf-client

grouping endpoints-container {
    description
        "This grouping is used to configure a set of NETCONF servers
        a NETCONF client may initiate connections to.";
    container endpoints {
        description
            "Container for the list of endpoints.";
        list endpoint {
            key name;
            unique "address port";
            min-elements 1;
            ordered-by user;
            description
                "A non-empty user-ordered list of endpoints for this NETCONF
                client to try to connect to. Defining more than one enables
                high-availability.";
        }
    }
}
```

```
    leaf name {
      type string;
      description
        "An arbitrary name for this endpoint.";
    }
    leaf address {
      type inet:host;
      mandatory true;
      description
        "The IP address or hostname of the endpoint.  If a
        hostname is configured and the DNS resolution results
        in more than one IP address, the NETCONF client
        will process the IP addresses as if they had been
        explicitly configured in place of the hostname.";
    }
    leaf port {
      type inet:port-number;
      description
        "The IP port for this endpoint.  The NETCONF client will
        use the IANA-assigned well-known port (set via a refine
        statement when uses) if no value is specified.";
    }
  }
}
```

<CODE ENDS>

### 3. 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.



## 3.1. Tree Diagram

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

```

module: ietf-netconf-server
  +--rw netconf-server
    +--rw session-options
    |   +--rw hello-timeout?  uint16
    +--rw listen {listen}?
    |   +--rw max-sessions?   uint16
    |   +--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 host-keys
    |                       +--rw host-key* [name]
    |                           +--rw name        string
    |                           +--rw (host-key-type)
    |                               +--:(public-key)
    |                                   +--rw public-key?
    |                                       -> /ks:keystore/keys/key/name
    |                               +--:(certificate)
    |                                   +--rw certificate?  leafref
    |                                       {sshcom:ssh-x509-certs}?
    |   +--rw client-cert-auth {sshcom:ssh-x509-certs}?
    |       +--rw trusted-ca-certs?      leafref
    |       +--rw trusted-client-certs?  leafref
    |   +--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
    |           +--rw compression
    |               +--rw compression-alg*  identityref
    |   +--:(tls) {tls-listen}?
    |       +--rw tls
    |           +--rw address?          inet:ip-address
    |           +--rw port?             inet:port-number
    |           +--rw certificates

```

```

|         |   +--rw certificate* [name]
|         |   |   +--rw name      leafref
+--rw client-auth
|   +--rw trusted-ca-certs?      leafref
|   +--rw trusted-client-certs?  leafref
|   +--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 (transport)
|   +--:(ssh) {ssh-call-home}?
|   |   +--rw ssh
|   |   |   +--rw endpoints
|   |   |   |   +--rw endpoint* [name]
|   |   |   |   |   +--rw name      string
|   |   |   |   |   +--rw address    inet:host
|   |   |   |   |   +--rw port?      inet:port-number
|   |   |   +--rw host-keys
|   |   |   |   +--rw host-key* [name]
|   |   |   |   |   +--rw name        string
|   |   |   |   |   +--rw (host-key-type)
|   |   |   |   |   |   +--:(public-key)
|   |   |   |   |   |   |   +--rw public-key?
|   |   |   |   |   |   |   |   -> /ks:keystore/keys/key/name
|   |   |   |   |   |   |   +--:(certificate)
|   |   |   |   |   |   |   |   +--rw certificate?  leafref
|   |   |   |   |   |   |   |   |   {sshcom:ssh-x509-certs}?
+--rw client-cert-auth {sshcom:ssh-x509-certs}?
|   +--rw trusted-ca-certs?      leafref
|   +--rw trusted-client-certs?  leafref
+--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
|         +--rw compression
|         |  +--rw compression-alg*  identityref
+---:(tls) {tls-call-home}?
+--rw tls
+--rw endpoints
|  +--rw endpoint* [name]
|  |  +--rw name      string
|  |  +--rw address   inet:host
|  |  +--rw port?     inet:port-number
+--rw certificates
|  +--rw certificate* [name]
|  |  +--rw name      leafref
+--rw client-auth
|  +--rw trusted-ca-certs?      leafref
|  +--rw trusted-client-certs? leafref
+--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 idle-timeout?  uint32
|  |  +--rw keep-alives
|  |  |  +--rw max-wait?      uint16
|  |  |  +--rw max-attempts? uint8
+---:(periodic-connection)
|  +--rw periodic!
|  |  +--rw idle-timeout?      uint16
|  |  +--rw reconnect-timeout? uint16
+--rw reconnect-strategy
+--rw start-with?      enumeration
+--rw max-attempts?    uint8

```

### 3.2. Example Usage

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

This example is consistent with the examples presented in Section 2.2 of [I-D.ietf-netconf-keystore].

```
<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">

  <!-- listening for SSH and TLS connections -->
  <listen>
    <endpoint> <!-- listening for SSH connections -->
      <name>netconf/ssh</name>
      <ssh>
        <address>11.22.33.44</address>
        <host-keys>
          <host-key>
            <name>public-key</name>
            <public-key>ex-rsa-key</public-key>
          </host-key>
          <host-key>
            <name>certificate</name>
            <certificate>builtin-idevid-cert</certificate>
          </host-key>
        </host-keys>
        <client-cert-auth>
          <trusted-ca-certs>deployment-specific-ca-certs</trusted-ca-certs>
          <trusted-client-certs>explicitly-trusted-client-certs</trusted-client-
certs>
        </client-cert-auth>
      </ssh>
    </endpoint>
    <endpoint> <!-- listening for TLS sessions -->
      <name>netconf/tls</name>
      <tls>
        <address>11.22.33.44</address>
        <certificates>
          <certificate>
            <name>tls-ec-cert</name>
          </certificate>
        </certificates>
        <client-auth>
          <trusted-ca-certs>deployment-specific-ca-certs</trusted-ca-certs>
          <trusted-client-certs>explicitly-trusted-client-certs</trusted-client-
certs>
```

```

    <cert-maps>
      <cert-to-name>
        <id>1</id>
        <fingerprint>11:0A:05:11:00</fingerprint>
        <map-type>x509c2n:san-any</map-type>
      </cert-to-name>
      <cert-to-name>
        <id>2</id>
        <fingerprint>B3:4F:A1:8C:54</fingerprint>
        <map-type>x509c2n:specified</map-type>
        <name>scooby-doo</name>
      </cert-to-name>
    </cert-maps>
  </client-auth>
</tls>
</endpoint>
</listen>

<!-- calling home to an SSH and TLS based NETCONF clients -->
<call-home>
  <netconf-client> <!-- SSH-based client -->
    <name>config-mgr</name>
    <ssh>
      <endpoints>
        <endpoint>
          <name>east-data-center</name>
          <address>11.22.33.44</address>
        </endpoint>
        <endpoint>
          <name>west-data-center</name>
          <address>55.66.77.88</address>
        </endpoint>
      </endpoints>
      <host-keys>
        <host-key>
          <name>certificate</name>
          <certificate>builtin-idevid-cert</certificate>
        </host-key>
      </host-keys>
      <client-cert-auth>
        <trusted-ca-certs>deployment-specific-ca-certs</trusted-ca-certs>
        <trusted-client-certs>explicitly-trusted-client-certs</trusted-client-
certs>
      </client-cert-auth>
    </ssh>
    <connection-type>
      <periodic>
        <idle-timeout>300</idle-timeout>
        <reconnect-timeout>60</reconnect-timeout>

```

```

    </periodic>
  </connection-type>
  <reconnect-strategy>
    <start-with>last-connected</start-with>
    <max-attempts>3</max-attempts>
  </reconnect-strategy>
</netconf-client>
<netconf-client> <!-- TLS-based client -->
  <name>event-correlator</name>
  <tls>
    <endpoints>
      <endpoint>
        <name>east-data-center</name>
        <address>22.33.44.55</address>
      </endpoint>
      <endpoint>
        <name>west-data-center</name>
        <address>33.44.55.66</address>
      </endpoint>
    </endpoints>
    <certificates>
      <certificate>
        <name>tls-ec-cert</name>
      </certificate>
    </certificates>
    <client-auth>
      <trusted-ca-certs>deployment-specific-ca-certs</trusted-ca-certs>
      <trusted-client-certs>explicitly-trusted-client-certs</trusted-client-
certs>
    <cert-maps>
      <cert-to-name>
        <id>1</id>
        <fingerprint>11:0A:05:11:00</fingerprint>
        <map-type>x509c2n:san-any</map-type>
      </cert-to-name>
      <cert-to-name>
        <id>2</id>
        <fingerprint>B3:4F:A1:8C:54</fingerprint>
        <map-type>x509c2n:specified</map-type>
        <name>scooby-doo</name>
      </cert-to-name>
    </cert-maps>
  </client-auth>
</tls>
<connection-type>
  <persistent>
    <idle-timeout>300</idle-timeout>
    <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>
```

### 3.3. YANG Model

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

```
<CODE BEGINS> file "ietf-netconf-server@2017-03-13.yang"

module ietf-netconf-server {
  yang-version 1.1;

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

  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 2017-03-13; // stable grouping definitions
    reference
      "RFC YYYY: SSH Client and Server Models";
  }

  import ietf-tls-server {
    prefix ts;
    revision-date 2017-03-13; // stable grouping definitions
    reference
```

```
    "RFC ZZZZ: TLS Client and Server Models";
}

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

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

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

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

  Copyright (c) 2014 IETF Trust and the persons identified as
  authors of the code. All rights reserved.

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

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

revision "2017-03-13" {
  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";
}

// top-level container (groupings below)
```

```
container netconf-server {
  description
    "Top-level container for NETCONF server configuration.";

  container session-options { // SHOULD WE REMOVE THIS ALTOGETHER?
    description
      "NETCONF session options, independent of transport
      or connection strategy.";
    leaf hello-timeout {
      type uint16;
      units "seconds";
      default 600;
      description
        "Specifies the maximum number of seconds that a SSH/TLS
        connection may wait for a hello message to be received.
        A connection will be dropped if no hello message is
        received before this number of seconds elapses. If set
        to zero, then the server will wait forever for a hello
        message.";
    }
  }
}

container listen {
  if-feature listen;
  description
    "Configures listen behavior";
  leaf max-sessions {
    type uint16;
    default 0;
    description
      "Specifies the maximum number of concurrent sessions
      that can be active at one time. The value 0 indicates
      that no artificial session limit should be used.";
  }
  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;
  }
}
```

```
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 of the interface to listen on. The
          SSH server will listen on all interfaces if no value
          is specified. Please note that some addresses have
          special meanings (e.g., '0.0.0.0' and ':::').";
      }
      leaf port {
        type inet:port-number;
        default 830;
        description
          "The local port number on this interface the SSH server
          listens on.";
      }
      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;
        description
          "The IP address of the interface to listen on. The
          TLS server will listen on all interfaces if no value
```



```
        refine endpoints/endpoint/port {
            default 4334;
        }
    }
    uses ss:ssh-server-grouping;
}

case tls {
    if-feature tls-call-home;
    container tls {
        description
            "Specifies TLS-specific call-home transport
            configuration.";
        uses endpoints-container {
            refine endpoints/endpoint/port {
                default 4335;
            }
        }
        uses ts:tls-server-grouping {
            augment "client-auth" {
                description
                    "Augments in the cert-to-name structure.";
                uses cert-maps-grouping;
            }
        }
    }
}

container connection-type {
    description
        "Indicates the kind of connection to use.";
    choice connection-type {
        description
            "Selects between available connection types.";
        case persistent-connection {
            container persistent {
                presence true;
                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.";
                leaf idle-timeout {
                    type uint32;
                }
            }
        }
    }
}
```

```

    units "seconds";
    default 86400; // one day;
    description
        "Specifies the maximum number of seconds that a
        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.";
}
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 3.1, item S6";
    leaf max-wait {
        type uint16 {
            range "1..max";
        }
        units seconds;
        default 30;
        description
            "Sets the amount of time in seconds after which
            if no data has been received from the 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 true;

```

```
description
    "Periodically connect to the NETCONF client, so that
     the NETCONF client may deliver messages pending for
     the NETCONF server. The NETCONF client must close
     the connection when it is ready to release it. Once
     the connection has been closed, the NETCONF server
     will restart its timer until the next connection.";
leaf idle-timeout {
    type uint16;
    units "seconds";
    default 300; // five minutes
    description
        "Specifies the maximum number of seconds that a
         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.";
}
leaf reconnect-timeout {
    type uint16 {
        range "1..max";
    }
    units minutes;
    default 60;
    description
        "Sets the maximum amount of unconnected time the
         NETCONF server will wait before re-establishing
         a connection to the NETCONF client. The NETCONF
         server may initiate a connection before this
         time if desired (e.g., to deliver an event
         notification message).";
}
}
}
}
}
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.";
        }
      }
      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).";
    }
  }
}

```

```

grouping cert-maps-grouping {
  description
    "A grouping that defines a container around the
     cert-to-name structure defined in RFC 7407.";
  container cert-maps {
    uses x509c2n:cert-to-name;
    description
      "The cert-maps container is used by a TLS-based NETCONF
       server to map the NETCONF client's presented X.509

```



```
        certificate to a NETCONF username.  If no matching and
        valid cert-to-name list entry can be found, then the
        NETCONF server MUST close the connection, and MUST NOT
        accept NETCONF messages over it.";
    reference
        "RFC WWW: NETCONF over TLS, Section 7";
}
}

grouping endpoints-container {
    description
        "This grouping is used to configure a set of NETCONF clients
        a NETCONF server may initiate call-home connections to.";
    container endpoints {
        description
            "Container for the list of endpoints.";
        list endpoint {
            key name;
            unique "address port";
            min-elements 1;
            ordered-by user;
            description
                "A non-empty user-ordered list of endpoints for this NETCONF
                server to try to connect to.  Defining more than one enables
                high-availability.";
            leaf name {
                type string;
                description
                    "An arbitrary name for this endpoint.";
            }
            leaf address {
                type inet:host;
                mandatory true;
                description
                    "The IP address or hostname of the endpoint.  If a
                    hostname is configured and the DNS resolution results
                    in more than one IP address, the NETCONF server
                    will process the IP addresses as if they had been
                    explicitly configured in place of the hostname.";
            }
            leaf port {
                type inet:port-number;
                description
                    "The IP port for this endpoint.  The NETCONF server will
                    use the IANA-assigned well-known port (set via a refine
                    statement when uses) if no value is specified.";
            }
        }
    }
}
```

```
    }  
  }  
}
```

<CODE ENDS>

#### 4. Design Considerations

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

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

##### 4.1. 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.

##### 4.2. 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.

##### 4.3. 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.

#### 4.4. 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.

#### 4.5. 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.

#### 4.6. For Call Home connections

The following objectives only pertain to call home connections.

##### 4.6.1. 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.

##### 4.6.2. 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.

##### 4.6.3. 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.

#### 4.6.4. 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.

#### 4.6.5. 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.

#### 4.6.6. 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.

#### 4.6.7. 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).

### 5. Security Considerations

A denial of service (DoS) attack MAY occur if the NETCONF server limits the maximum number of NETCONF sessions it will accept (i.e. the 'max-sessions' field in the ietf-netconf-server module is not zero) and either the "hello-timeout" or "idle-timeout" fields in

ietf-netconf-server module have been set to indicate the NETCONF server should wait forever (i.e. set to zero).

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) [RFC6536] 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:

NONE

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

## 6. IANA Considerations

### 6.1. The IETF XML Registry

This document registers two URIs in 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.

### 6.2. The YANG Module Names Registry

This document registers two YANG modules in the YANG Module Names registry [RFC7950]. Following the format in [RFC7950], 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

## 7. Acknowledgements

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

Juergen Schoenwaelder and was partly funded by Flamingo, a Network of Excellence project (ICT-318488) supported by the European Commission under its Seventh Framework Programme.

## 8. References

### 8.1. Normative References

- [I-D.ietf-netconf-keystore]  
Watsen, K. and G. Wu, "Keystore Model", draft-ietf-netconf-keystore-00 (work in progress), October 2016.
- [I-D.ietf-netconf-ssh-client-server]  
Watsen, K. and G. Wu, "SSH Client and Server Models", draft-ietf-netconf-ssh-client-server-01 (work in progress), November 2016.
- [I-D.ietf-netconf-tls-client-server]  
Watsen, K., "TLS Client and Server Models", draft-ietf-netconf-tls-client-server-01 (work in progress), November 2016.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [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, <<http://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, <<http://www.rfc-editor.org/info/rfc6242>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<http://www.rfc-editor.org/info/rfc6991>>.
- [RFC7407] Bjorklund, M. and J. Schoenwaelder, "A YANG Data Model for SNMP Configuration", RFC 7407, DOI 10.17487/RFC7407, December 2014, <<http://www.rfc-editor.org/info/rfc7407>>.
- [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, <<http://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, <<http://www.rfc-editor.org/info/rfc7950>>.

## 8.2. Informative References

- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<http://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, <<http://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, <<http://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, <<http://www.rfc-editor.org/info/rfc4254>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<http://www.rfc-editor.org/info/rfc5246>>.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <<http://www.rfc-editor.org/info/rfc6536>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<http://www.rfc-editor.org/info/rfc8040>>.
- [RFC8071] Watsen, K., "NETCONF Call Home and RESTCONF Call Home", RFC 8071, DOI 10.17487/RFC8071, February 2017, <<http://www.rfc-editor.org/info/rfc8071>>.



## Appendix A. Change Log

## A.1. server-model-09 to 00

- o This draft was split out from draft-ietf-netconf-server-model-09.
- o Added in previously missing ietf-netconf-client module.
- o Added in new features 'listen' and 'call-home' so future transports can be augmented in.

## A.2. 00 to 01

- o Renamed "keychain" to "keystore".

## A.3. 01 to 02

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

## Appendix B. Open Issues

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

## Authors' Addresses

Kent Watsen  
Juniper Networks

EMail: [kwatsen@juniper.net](mailto:kwatsen@juniper.net)

Gary Wu  
Cisco Networks

EMail: [garywu@cisco.com](mailto:garywu@cisco.com)

Juergen Schoenwaelder  
Jacobs University Bremen

EMail: [j.schoenwaelder@jacobs-university.de](mailto:j.schoenwaelder@jacobs-university.de)

NETCONF Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: 8 September 2022

K. Watsen  
Watsen Networks  
7 March 2022

NETCONF Client and Server Models  
draft-ietf-netconf-netconf-client-server-25

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

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

- \* AAAA --> the assigned RFC value for draft-ietf-netconf-crypto-types
- \* BBBB --> the assigned RFC value for draft-ietf-netconf-trust-anchors
- \* CCCC --> the assigned RFC value for draft-ietf-netconf-keystore
- \* DDDD --> the assigned RFC value for draft-ietf-netconf-tcp-client-server
- \* EEEE --> the assigned RFC value for draft-ietf-netconf-ssh-client-server
- \* FFFF --> the assigned RFC value for draft-ietf-netconf-tls-client-server
- \* GGGG --> the assigned RFC value for draft-ietf-netconf-http-client-server
- \* HHHH --> 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:

\* 2022-03-07 --> the publication date of this draft

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

\* Appendix A. Change Log

#### Status of This Memo

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

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

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

This Internet-Draft will expire on 8 September 2022.

#### Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

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

#### Table of Contents

1. Introduction . . . . .	4
1.1. Relation to other RFCs . . . . .	4
1.2. Specification Language . . . . .	5
1.3. Adherence to the NMDA . . . . .	5
1.4. Conventions . . . . .	5
2. The "ietf-netconf-client" Module . . . . .	6

2.1.	Data Model Overview . . . . .	6
2.2.	Example Usage . . . . .	10
2.3.	YANG Module . . . . .	14
3.	The "ietf-netconf-server" Module . . . . .	25
3.1.	Data Model Overview . . . . .	25
3.2.	Example Usage . . . . .	30
3.3.	YANG Module . . . . .	36
4.	Security Considerations . . . . .	50
4.1.	The "ietf-netconf-client" YANG Module . . . . .	50
4.2.	The "ietf-netconf-server" YANG Module . . . . .	51
5.	IANA Considerations . . . . .	51
5.1.	The "IETF XML" Registry . . . . .	51
5.2.	The "YANG Module Names" Registry . . . . .	52
6.	References . . . . .	52
6.1.	Normative References . . . . .	52
6.2.	Informative References . . . . .	53
Appendix A.	Change Log . . . . .	55
A.1.	00 to 01 . . . . .	55
A.2.	01 to 02 . . . . .	55
A.3.	02 to 03 . . . . .	55
A.4.	03 to 04 . . . . .	55
A.5.	04 to 05 . . . . .	56
A.6.	05 to 06 . . . . .	56
A.7.	06 to 07 . . . . .	56
A.8.	07 to 08 . . . . .	56
A.9.	08 to 09 . . . . .	56
A.10.	09 to 10 . . . . .	57
A.11.	10 to 11 . . . . .	57
A.12.	11 to 12 . . . . .	57
A.13.	12 to 13 . . . . .	57
A.14.	13 to 14 . . . . .	58
A.15.	14 to 15 . . . . .	58
A.16.	15 to 16 . . . . .	58
A.17.	16 to 17 . . . . .	58
A.18.	17 to 18 . . . . .	58
A.19.	18 to 19 . . . . .	58
A.20.	19 to 20 . . . . .	59
A.21.	20 to 21 . . . . .	59
A.22.	21 to 22 . . . . .	59
A.23.	22 to 23 . . . . .	59
A.24.	23 to 24 . . . . .	59
A.25.	24 to 25 . . . . .	59
Acknowledgements	. . . . .	60
Author's Address	. . . . .	60

## 1. Introduction

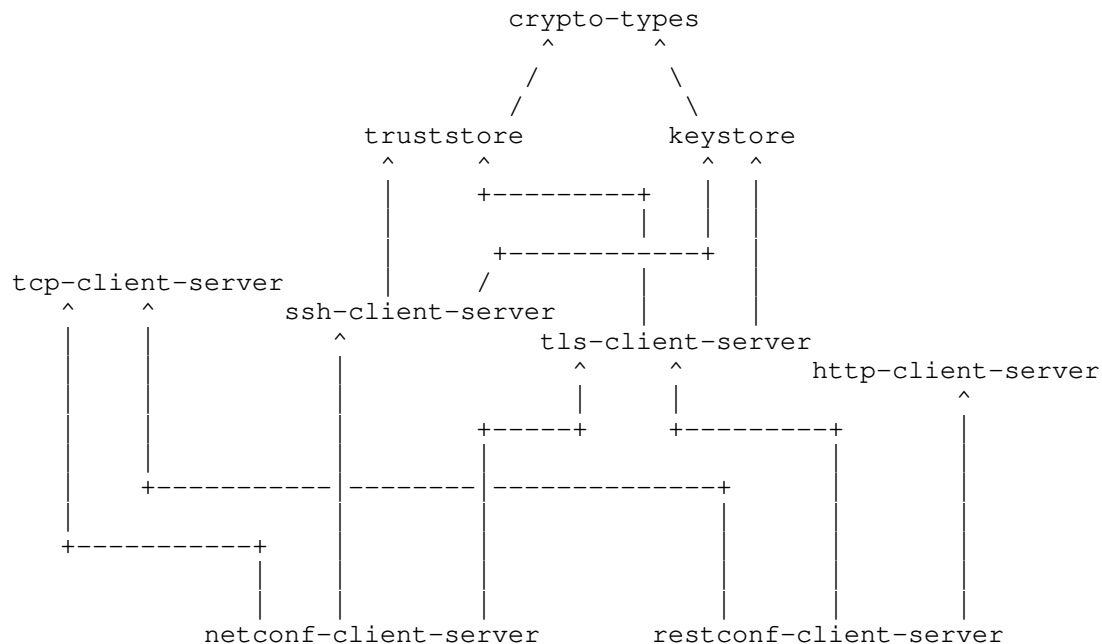
This document defines two YANG [RFC7950] modules, one module to configure a NETCONF [RFC6241] client and the other module to configure a NETCONF server. Both modules support both NETCONF over SSH [RFC6242] and NETCONF over TLS [RFC7589] and NETCONF Call Home connections [RFC8071].

### 1.1. Relation to other RFCs

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

The modules have been defined in a modular fashion to enable their use by other efforts, some of which are known to be in progress at the time of this writing, with many more expected to be defined in time.

The normative dependency relationship between the various RFCs in the collection is presented in the below diagram. The labels in the diagram represent the primary purpose provided by each RFC. Hyperlinks to each RFC are provided below the diagram.



Label in Diagram	Originating RFC
crypto-types	[I-D.ietf-netconf-crypto-types]
truststore	[I-D.ietf-netconf-trust-anchors]
keystore	[I-D.ietf-netconf-keystore]
tcp-client-server	[I-D.ietf-netconf-tcp-client-server]
ssh-client-server	[I-D.ietf-netconf-ssh-client-server]
tls-client-server	[I-D.ietf-netconf-tls-client-server]
http-client-server	[I-D.ietf-netconf-http-client-server]
netconf-client-server	[I-D.ietf-netconf-netconf-client-server]
restconf-client-server	[I-D.ietf-netconf-restconf-client-server]

Table 1: Label to RFC Mapping

## 1.2. Specification Language

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

## 1.3. Adherence to the NMDA

This document is compliant with the Network Management Datastore Architecture (NMDA) [RFC8342]. For instance, as described in [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore], trust anchors and keys installed during manufacturing are expected to appear in <operational>.

## 1.4. Conventions

Various examples used in this document use a placeholder value for binary data that has been base64 encoded (e.g., "BASE64VALUE="). This placeholder value is used as real base64 encoded structures are often many lines long and hence distracting to the example being presented.

## 2. The "ietf-netconf-client" Module

The NETCONF client model presented in this section supports both clients initiating connections to servers, as well as clients listening for connections from servers calling home, using either the SSH and TLS transport protocols.

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

### 2.1. Data Model Overview

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

#### 2.1.1. Features

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

Features:

```
+-- ssh-initiate
+-- tls-initiate
+-- ssh-listen
+-- tls-listen
```

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

#### 2.1.2. Groupings

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

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

Each of these groupings are presented in the following subsections.

##### 2.1.2.1. The "netconf-client-grouping" Grouping

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

```
grouping netconf-client-grouping ---> <empty>
```



## Comments:

- \* This grouping does not define any nodes, but is maintained so that downstream modules can augment nodes into it if needed.
- \* The "netconf-client-grouping" defines, if it can be called that, the configuration for just "NETCONF" part of a protocol stack. It does not, for instance, define any configuration for the "TCP", "SSH" or "TLS" protocol layers (for that, see Section 2.1.2.2 and Section 2.1.2.3).

## 2.1.2.2. The "netconf-client-initiate-stack-grouping" Grouping

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

```

grouping netconf-client-initiate-stack-grouping
  +-- (transport)
    +--:(ssh) {ssh-initiate}?
      +-- ssh
        +-- tcp-client-parameters
          | +---u tcpc:tcp-client-grouping
        +-- ssh-client-parameters
          | +---u sshc:ssh-client-grouping
        +-- netconf-client-parameters
          +---u ncc:netconf-client-grouping
    +--:(tls) {tls-initiate}?
      +-- tls
        +-- tcp-client-parameters
          | +---u tcpc:tcp-client-grouping
        +-- tls-client-parameters
          | +---u tlsc:tls-client-grouping
        +-- netconf-client-parameters
          +---u ncc:netconf-client-grouping
  
```

## Comments:

- \* The "netconf-client-initiate-stack-grouping" defines the configuration for a full NETCONF protocol stack, for NETCONF clients that initiate connections to NETCONF servers, as opposed to receiving call-home [RFC8071] connections.
- \* The "transport" choice node enables both the SSH and TLS transports to be configured, with each option enabled by a "feature" statement.
- \* For the referenced grouping statement(s):

- The "tcp-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
- The "ssh-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-ssh-client-server].
- The "tls-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tls-client-server].
- The "netconf-client-grouping" grouping is discussed in Section 2.1.2.1 in this document.

#### 2.1.2.3. The "netconf-client-listen-stack-grouping" Grouping

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

```

grouping netconf-client-listen-stack-grouping
+-- (transport)
+--:(ssh) {ssh-listen}?
|   +-- ssh
|       +-- tcp-server-parameters
|           | +---u tcps:tcp-server-grouping
|       +-- ssh-client-parameters
|           | +---u sshc:ssh-client-grouping
|       +-- netconf-client-parameters
|           +---u ncc:netconf-client-grouping
+--:(tls) {tls-listen}?
|   +-- tls
|       +-- tcp-server-parameters
|           | +---u tcps:tcp-server-grouping
|       +-- tls-client-parameters
|           | +---u tlsc:tls-client-grouping
|       +-- netconf-client-parameters
|           +---u ncc:netconf-client-grouping

```

#### Comments:

- \* The "netconf-client-listen-stack-grouping" defines the configuration for a full NETCONF protocol stack, for NETCONF clients that receive call-home [RFC8071] connections from NETCONF servers.
- \* The "transport" choice node enables both the SSH and TLS transports to be configured, with each option enabled by a "feature" statement.
- \* For the referenced grouping statement(s):
  - The "tcp-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tcp-client-server].

- The "ssh-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-ssh-client-server].
- The "tls-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tls-client-server].
- The "netconf-client-grouping" grouping is discussed in Section 2.1.2.1 in this document.

#### 2.1.2.4. The "netconf-client-app-grouping" Grouping

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

```

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

Comments:

- \* The "netconf-client-app-grouping" defines the configuration for a NETCONF client that supports both initiating connections to NETCONF servers as well as receiving call-home connections from NETCONF servers.
- \* Both the "initiate" and "listen" subtrees must be enabled by "feature" statements.

- \* For the referenced grouping statement(s):
  - The "netconf-client-initiate-stack-grouping" grouping is discussed in Section 2.1.2.2 in this document.
  - The "netconf-client-listen-stack-grouping" grouping is discussed in Section 2.1.2.3 in this document.

### 2.1.3. Protocol-accessible Nodes

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

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

Comments:

- \* Protocol-accessible nodes are those nodes that are accessible when the module is "implemented", as described in Section 5.6.5 of [RFC7950].
- \* For the "ietf-netconf-client" module, the protocol-accessible nodes are an instance of the "netconf-client-app-grouping" discussed in Section 2.1.2.4 grouping.
- \* The reason for why "netconf-client-app-grouping" exists separate from the protocol-accessible nodes definition is so as to enable instances of netconf-client-app-grouping to be instantiated in other locations, as may be needed or desired by some modules.

### 2.2. Example Usage

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

This example is consistent with the examples presented in Section 2.2 of [I-D.ietf-netconf-trust-anchors] and Section 2.2 of [I-D.ietf-netconf-keystore].

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

```
<netconf-client xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-clie\
nt">
```

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

```

<initiate>
  <netconf-server>
    <name>corp-fw1</name>
    <endpoints>
      <endpoint>
        <name>corp-fw1.example.com</name>
        <ssh>
          <tcp-client-parameters>
            <remote-address>corp-fw1.example.com</remote-address>
            <keepalives>
              <idle-time>15</idle-time>
              <max-probes>3</max-probes>
              <probe-interval>30</probe-interval>
            </keepalives>
          </tcp-client-parameters>
          <ssh-client-parameters>
            <client-identity>
              <username>foobar</username>
              <public-key>
                <keystore-reference>ssh-rsa-key</keystore-referenc\
e>
              </public-key>
            </client-identity>
            <server-authentication>
              <ca-certs>
                <truststore-reference>trusted-server-ca-certs</tru\
ststore-reference>
              </ca-certs>
              <ee-certs>
                <truststore-reference>trusted-server-ee-certs</tru\
ststore-reference>
              </ee-certs>
            </server-authentication>
            <keepalives>
              <max-wait>30</max-wait>
              <max-attempts>3</max-attempts>
            </keepalives>
          </ssh-client-parameters>
          <netconf-client-parameters>
            <!-- nothing to configure -->
          </netconf-client-parameters>
        </ssh>
      </endpoint>
    </endpoints>
    <name>corp-fw2.example.com</name>
    <tls>
      <tcp-client-parameters>
        <remote-address>corp-fw2.example.com</remote-address>

```

```

        <keepalives>
          <idle-time>15</idle-time>
          <max-probes>3</max-probes>
          <probe-interval>30</probe-interval>
        </keepalives>
      </tcp-client-parameters>
      <tls-client-parameters>
        <client-identity>
          <certificate>
            <keystore-reference>
              <asymmetric-key>rsa-asymmetric-key</asymmetric-k\
ey>
              <certificate>ex-rsa-cert</certificate>
            </keystore-reference>
          </certificate>
        </client-identity>
        <server-authentication>
          <ca-certs>
            <truststore-reference>trusted-server-ca-certs</tru\
ststore-reference>
          </ca-certs>
          <ee-certs>
            <truststore-reference>trusted-server-ee-certs</tru\
ststore-reference>
          </ee-certs>
        </server-authentication>
      </keepalives>
      <test-peer-aliveness>
        <max-wait>30</max-wait>
        <max-attempts>3</max-attempts>
      </test-peer-aliveness>
    </keepalives>
  </tls-client-parameters>
  <netconf-client-parameters>
    <!-- nothing to configure -->
  </netconf-client-parameters>
</tls>
</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 SSH listener</name>
    <ssh>
      <tcp-server-parameters>
        <local-address>192.0.2.7</local-address>
      </tcp-server-parameters>
      <ssh-client-parameters>
        <client-identity>
          <username>foobar</username>
          <public-key>
            <keystore-reference>ssh-rsa-key</keystore-reference>
          </public-key>
        </client-identity>
        <server-authentication>
          <ca-certs>
            <truststore-reference>trusted-server-ca-certs</truststore-reference>
          </ca-certs>
          <ee-certs>
            <truststore-reference>trusted-server-ee-certs</truststore-reference>
          </ee-certs>
          <ssh-host-keys>
            <truststore-reference>trusted-ssh-public-keys</truststore-reference>
          </ssh-host-keys>
        </server-authentication>
      </ssh-client-parameters>
      <netconf-client-parameters>
        <!-- nothing to configure -->
      </netconf-client-parameters>
    </ssh>
  </endpoint>
  <endpoint>
    <name>Intranet-facing TLS listener</name>
    <tls>
      <tcp-server-parameters>
        <local-address>192.0.2.7</local-address>
      </tcp-server-parameters>
      <tls-client-parameters>
        <client-identity>
          <certificate>
            <keystore-reference>
              <asymmetric-key>rsa-asymmetric-key</asymmetric-key>
              <certificate>ex-rsa-cert</certificate>
            </keystore-reference>
          </certificate>
        </client-identity>
      </tls-client-parameters>
    </tls>
  </endpoint>

```

```

        </certificate>
      </client-identity>
    <server-authentication>
      <ca-certs>
        <truststore-reference>trusted-server-ca-certs</truststore-reference>
      </ca-certs>
      <ee-certs>
        <truststore-reference>trusted-server-ee-certs</truststore-reference>
      </ee-certs>
    </server-authentication>
    <keepalives>
      <peer-allowed-to-send/>
    </keepalives>
  </tls-client-parameters>
  <netconf-client-parameters>
    <!-- nothing to configure -->
  </netconf-client-parameters>
</tls>
</endpoint>
</listen>
</netconf-client>

```

### 2.3. YANG Module

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

```
<CODE BEGINS> file "ietf-netconf-client@2022-03-07.yang"
```

```

module ietf-netconf-client {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-netconf-client";
  prefix ncc;

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

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

```



```
}

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

import ietf-ssh-client {
  prefix sshc;
  revision-date 2022-03-07; // stable grouping definitions
  reference
    "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
}

import ietf-tls-client {
  prefix tlsc;
  revision-date 2022-03-07; // stable grouping definitions
  reference
    "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
}

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

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

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

  Copyright (c) 2021 IETF Trust and the persons identified
  as authors of the code. All rights reserved.

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

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

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

```
revision 2022-03-07 {
  description
    "Initial version";
  reference
    "RFC HHHH: NETCONF Client and Server Models";
}

// Features

feature ssh-initiate {
  description
    "The 'ssh-initiate' feature indicates that the NETCONF client
    supports initiating SSH connections to NETCONF servers.";
  reference
    "RFC 6242:
      Using the NETCONF Protocol over Secure Shell (SSH)";
}

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

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

feature tls-listen {
  description
    "The 'tls-listen' feature indicates that the NETCONF client
    supports opening a port to listen for incoming NETCONF
    server call-home TLS connections.";
```

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

// Groupings

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

    This grouping currently does not define any nodes.";
}

grouping netconf-client-initiate-stack-grouping {
  description
    "A reusable grouping for configuring a NETCONF client
    'initiate' protocol stack for a single connection.";
  choice transport {
    mandatory true;
    description
      "Selects between available transports.";
    case ssh {
      if-feature "ssh-initiate";
      container ssh {
        description
          "Specifies IP and SSH specific configuration
          for the connection.";
        container tcp-client-parameters {
          description
            "A wrapper around the TCP client parameters
            to avoid name collisions.";
          uses tcpc:tcp-client-grouping {
            refine "remote-port" {
              default "830";
              description
                "The NETCONF client will attempt to connect
                to the IANA-assigned well-known port value
                for 'netconf-ssh' (830) if no value is
                specified.";
            }
          }
        }
      }
    }
    container ssh-client-parameters {
      description
        "A wrapper around the SSH client parameters to
        avoid name collisions.";
    }
  }
}
```

```
        uses sshc:ssh-client-grouping;
    }
    container netconf-client-parameters {
        description
            "A wrapper around the NETCONF client parameters
            to avoid name collisions.";
        uses ncc:netconf-client-grouping;
    }
}
case tls {
    if-feature "tls-initiate";
    container tls {
        description
            "Specifies IP and TLS specific configuration
            for the connection.";
        container tcp-client-parameters {
            description
                "A wrapper around the TCP client parameters
                to avoid name collisions.";
            uses tcpc:tcp-client-grouping {
                refine "remote-port" {
                    default "6513";
                    description
                        "The NETCONF client will attempt to connect
                        to the IANA-assigned well-known port value
                        for 'netconf-tls' (6513) if no value is
                        specified.";
                }
            }
        }
    }
    container tls-client-parameters {
        must client-identity {
            description
                "NETCONF/TLS clients MUST pass some
                authentication credentials.";
        }
        description
            "A wrapper around the TLS client parameters
            to avoid name collisions.";
        uses tlsc:tls-client-grouping;
    }
    container netconf-client-parameters {
        description
            "A wrapper around the NETCONF client parameters
            to avoid name collisions.";
        uses ncc:netconf-client-grouping;
    }
}
```

```
    }
  }
}
} // netconf-client-initiate-stack-grouping

grouping netconf-client-listen-stack-grouping {
  description
    "A reusable grouping for configuring a NETCONF client
    'listen' protocol stack for a single connection. The
    'listen' stack supports call home connections, as
    described in RFC 8071";
  reference
    "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
  choice transport {
    mandatory true;
    description
      "Selects between available transports.";
    case ssh {
      if-feature "ssh-listen";
      container ssh {
        description
          "SSH-specific listening configuration for inbound
          connections.";
        container tcp-server-parameters {
          description
            "A wrapper around the TCP server parameters
            to avoid name collisions.";
          uses tcps:tcp-server-grouping {
            refine "local-port" {
              default "4334";
              description
                "The NETCONF client will listen on the IANA-
                assigned well-known port for 'netconf-ch-ssh'
                (4334) if no value is specified.";
            }
          }
        }
      }
    }
    container ssh-client-parameters {
      description
        "A wrapper around the SSH client parameters
        to avoid name collisions.";
      uses sshc:ssh-client-grouping;
    }
    container netconf-client-parameters {
      description
        "A wrapper around the NETCONF client parameters
        to avoid name collisions.";
      uses ncc:netconf-client-grouping;
    }
  }
}
```

```
    }
  }
}
case tls {
  if-feature "tls-listen";
  container tls {
    description
      "TLS-specific listening configuration for inbound
      connections.";
    container tcp-server-parameters {
      description
        "A wrapper around the TCP server parameters
        to avoid name collisions.";
      uses tcps:tcp-server-grouping {
        refine "local-port" {
          default "4334";
          description
            "The NETCONF client will listen on the IANA-
            assigned well-known port for 'netconf-ch-ssh'
            (4334) if no value is specified.";
        }
      }
    }
    container tls-client-parameters {
      must client-identity {
        description
          "NETCONF/TLS clients MUST pass some
          authentication credentials.";
      }
      description
        "A wrapper around the TLS client parameters
        to avoid name collisions.";
      uses tlsc:tls-client-grouping;
    }
    container netconf-client-parameters {
      description
        "A wrapper around the NETCONF client parameters
        to avoid name collisions.";
      uses ncc:netconf-client-grouping;
    }
  }
}
} // netconf-client-listen-stack-grouping

grouping netconf-client-app-grouping {
  description
    "A reusable grouping for configuring a NETCONF client
```

```
    application that supports both 'initiate' and 'listen'
    protocol stacks for a multiplicity of connections.";
  container initiate {
    if-feature "ssh-initiate or tls-initiate";
    presence
      "Indicates that client-initiated connections have been
      configured. This statement is present so the mandatory
      descendant nodes do not imply that this node must be
      configured.";
    description
      "Configures client initiating underlying TCP connections.";
    list netconf-server {
      key "name";
      min-elements 1;
      description
        "List of NETCONF servers the NETCONF client is to
        maintain simultaneous connections with.";
      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.";
          }
          uses netconf-client-initiate-stack-grouping;
        } // list endpoint
      } // container endpoints

      container connection-type {
        description
          "Indicates the NETCONF client's preference for how the
          NETCONF connection is maintained.";
        choice connection-type {
```

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

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

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

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

      In the case that the previous connection is still
      active, establishing a new connection is NOT
      RECOMMENDED.";
    leaf period {
      type uint16;
      units "minutes";
      default "60";
      description
        "Duration of time between periodic connections.";
    }
    leaf anchor-time {
      type yang:date-and-time {
        // constrained to minute-level granularity
        pattern '\d{4}-\d{2}-\d{2}T\d{2}:\d{2}'
          + '(Z|[\+|-]\d{2}:\d{2})';
      }
    }
  }
}
```



```

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

```

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

container listen {
    if-feature "ssh-listen or tls-listen";
    presence
        "Indicates that client-listening ports have been configured.
        This statement is present so the mandatory descendant nodes
        do not imply that this node must be configured.";
    description
        "Configures the client to accept call-home TCP connections.";
    leaf idle-timeout {
        type uint16;
        units "seconds";
        default "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.";
        }
        uses netconf-client-listen-stack-grouping;
    } // endpoint
} // listen
} // netconf-client-app-grouping

// Protocol accessible node for clients that implement this module.
container netconf-client {
    uses netconf-client-app-grouping;
    description
        "Top-level container for NETCONF client configuration.";
}
}
```

<CODE ENDS>

### 3. The "ietf-netconf-server" Module

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

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

#### 3.1. Data Model Overview

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

##### 3.1.1. Features

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

## Features:

```
+-- ssh-listen
+-- tls-listen
+-- ssh-call-home
+-- tls-call-home
```

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

## 3.1.2. Groupings

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

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

Each of these groupings are presented in the following subsections.

## 3.1.2.1. The "netconf-server-grouping" Grouping

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

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

## Comments:

- \* The "netconf-server-grouping" defines the configuration for just "NETCONF" part of a protocol stack. It does not, for instance, define any configuration for the "TCP", "SSH" or "TLS" protocol layers (for that, see Section 3.1.2.2 and Section 3.1.2.3).
- \* The "client-identity-mappings" node, which must be enabled by "feature" statements, defines a mapping from certificate fields to NETCONF user names.
- \* For the referenced grouping statement(s):
  - The "cert-to-name" grouping is discussed in Section 4.1 of [RFC7407].

### 3.1.2.2. The "netconf-server-listen-stack-grouping" Grouping

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

```

grouping netconf-server-listen-stack-grouping
  +-- (transport)
    +--:(ssh) {ssh-listen}?
      +-- ssh
        +-- tcp-server-parameters
          | +---u tcps:tcp-server-grouping
        +-- ssh-server-parameters
          | +---u sshs:ssh-server-grouping
        +-- netconf-server-parameters
          | +---u ncs:netconf-server-grouping
    +--:(tls) {tls-listen}?
      +-- tls
        +-- tcp-server-parameters
          | +---u tcps:tcp-server-grouping
        +-- tls-server-parameters
          | +---u tlss:tls-server-grouping
        +-- netconf-server-parameters
          | +---u ncs:netconf-server-grouping

```

#### Comments:

- \* The "netconf-server-listen-stack-grouping" defines the configuration for a full NETCONF protocol stack for NETCONF servers that listen for standard connections from NETCONF clients, as opposed to initiating call-home [RFC8071] connections.
- \* The "transport" choice node enables both the SSH and TLS transports to be configured, with each option enabled by a "feature" statement.
- \* For the referenced grouping statement(s):
  - The "tcp-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
  - The "ssh-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-ssh-client-server].
  - The "tls-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tls-client-server].
  - The "netconf-server-grouping" is discussed in Section 3.1.2.1 of this document.

### 3.1.2.3. The "netconf-server-callhome-stack-grouping" Grouping

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

```

grouping netconf-server-callhome-stack-grouping
  +-- (transport)
    +--:(ssh) {ssh-call-home}?
      +-- ssh
        +-- tcp-client-parameters
          | +---u tcpc:tcp-client-grouping
        +-- ssh-server-parameters
          | +---u sshs:ssh-server-grouping
        +-- netconf-server-parameters
          | +---u ncs:netconf-server-grouping
    +--:(tls) {tls-call-home}?
      +-- tls
        +-- tcp-client-parameters
          | +---u tcpc:tcp-client-grouping
        +-- tls-server-parameters
          | +---u tlss:tls-server-grouping
        +-- netconf-server-parameters
          | +---u ncs:netconf-server-grouping

```

Comments:

- \* The "netconf-server-callhome-stack-grouping" defines the configuration for a full NETCONF protocol stack, for NETCONF servers that initiate call-home [RFC8071] connections to NETCONF clients.
- \* The "transport" choice node enables both the SSH and TLS transports to be configured, with each option enabled by a "feature" statement.
- \* For the referenced grouping statement(s):
  - The "tcp-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
  - The "ssh-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-ssh-client-server].
  - The "tls-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tls-client-server].
  - The "netconf-server-grouping" is discussed in Section 3.1.2.1 of this document.

### 3.1.2.4. The "netconf-server-app-grouping" Grouping

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

```

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

Comments:

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

### 3.1.3. Protocol-accessible Nodes

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

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

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

Comments:

- \* Protocol-accessible nodes are those nodes that are accessible when the module is "implemented", as described in Section 5.6.5 of [RFC7950].
- \* For the "ietf-netconf-server" module, the protocol-accessible nodes are an instance of the "netconf-server-app-grouping" discussed in Section 3.1.2.4 grouping.
- \* The reason for why "netconf-server-app-grouping" exists separate from the protocol-accessible nodes definition is so as to enable instances of netconf-server-app-grouping to be instantiated in other locations, as may be needed or desired by some modules.

### 3.2. Example Usage

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

This example is consistent with the examples presented in Section 2.2 of [I-D.ietf-netconf-trust-anchors] and Section 2.2 of [I-D.ietf-netconf-keystore].

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

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

```
  <!-- endpoints to listen for NETCONF connections on -->
  <listen>
    <endpoint> <!-- listening for SSH connections -->
      <name>netconf/ssh</name>
      <ssh>
```



```

    <tcp-server-parameters>
      <local-address>192.0.2.7</local-address>
    </tcp-server-parameters>
    <ssh-server-parameters>
      <server-identity>
        <host-key>
          <name>deployment-specific-certificate</name>
          <public-key>
            <keystore-reference>ssh-rsa-key</keystore-reference>
          </public-key>
        </host-key>
      </server-identity>
      <client-authentication>
      </client-authentication>
    </ssh-server-parameters>
    <netconf-server-parameters>
      <!-- nothing to configure -->
    </netconf-server-parameters>
  </ssh>
</endpoint>
<endpoint> <!-- listening for TLS sessions -->
  <name>netconf/tls</name>
  <tls>
    <tcp-server-parameters>
      <local-address>192.0.2.7</local-address>
    </tcp-server-parameters>
    <tls-server-parameters>
      <server-identity>
        <certificate>
          <keystore-reference>
            <asymmetric-key>rsa-asymmetric-key</asymmetric-key>
            <certificate>ex-rsa-cert</certificate>
          </keystore-reference>
        </certificate>
      </server-identity>
      <client-authentication>
        <ca-certs>
          <truststore-reference>trusted-client-ca-certs</truststore-reference>
        </ca-certs>
        <ee-certs>
          <truststore-reference>trusted-client-ee-certs</truststore-reference>
        </ee-certs>
      </client-authentication>
      <keepalives>
        <peer-allowed-to-send/>
      </keepalives>

```

```

    </tls-server-parameters>
    <netconf-server-parameters>
      <client-identity-mappings>
        <cert-to-name>
          <id>1</id>
          <fingerprint>11:0A:05:11:00</fingerprint>
          <map-type>x509c2n:specified</map-type>
          <name>scooby-doo</name>
        </cert-to-name>
        <cert-to-name>
          <id>2</id>
          <map-type>x509c2n:san-any</map-type>
        </cert-to-name>
      </client-identity-mappings>
    </netconf-server-parameters>
  </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>
          <tcp-client-parameters>
            <remote-address>east.config-mgr.example.com</remote-ad\
dress>

            <keepalives>
              <idle-time>15</idle-time>
              <max-probes>3</max-probes>
              <probe-interval>30</probe-interval>
            </keepalives>
          </tcp-client-parameters>
          <ssh-server-parameters>
            <server-identity>
              <host-key>
                <name>deployment-specific-certificate</name>
                <public-key>
                  <keystore-reference>ssh-rsa-key</keystore-refere\
nce>

                </public-key>
              </host-key>
            </server-identity>
          </ssh-server-parameters>
        </netconf-server-parameters>

```

```

        <!-- nothing to configure -->
      </netconf-server-parameters>
    </ssh>
  </endpoint>
</endpoints>
  <name>west-data-center</name>
  <ssh>
    <tcp-client-parameters>
      <remote-address>west.config-mgr.example.com</remote-ad\
dress>
    </tcp-client-parameters>
    <ssh-server-parameters>
      <server-identity>
        <host-key>
          <name>deployment-specific-certificate</name>
          <public-key>
            <keystore-reference>ssh-rsa-key</keystore-refere\
nce>
          </public-key>
        </host-key>
      </server-identity>
    </ssh-server-parameters>
  </netconf-server-parameters>
  <!-- nothing to configure -->
</netconf-server-parameters>
</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>
        <tcp-client-parameters>
          <remote-address>east.analytics.example.com</remote-add\
ress>

```

```

    <keepalives>
      <idle-time>15</idle-time>
      <max-probes>3</max-probes>
      <probe-interval>30</probe-interval>
    </keepalives>
  </tcp-client-parameters>
  <tls-server-parameters>
    <server-identity>
      <certificate>
        <keystore-reference>
          <asymmetric-key>rsa-asymmetric-key</asymmetric-k\
ey>
          <certificate>ex-rsa-cert</certificate>
        </keystore-reference>
      </certificate>
    </server-identity>
    <client-authentication>
      <ca-certs>
        <truststore-reference>trusted-client-ca-certs</tru\
ststore-reference>
      </ca-certs>
      <ee-certs>
        <truststore-reference>trusted-client-ee-certs</tru\
ststore-reference>
      </ee-certs>
    </client-authentication>
  </keepalives>
  <test-peer-aliveness>
    <max-wait>30</max-wait>
    <max-attempts>3</max-attempts>
  </test-peer-aliveness>
</keepalives>
</tls-server-parameters>
<netconf-server-parameters>
  <client-identity-mappings>
    <cert-to-name>
      <id>1</id>
      <fingerprint>11:0A:05:11:00</fingerprint>
      <map-type>x509c2n:specified</map-type>
      <name>scooby-doo</name>
    </cert-to-name>
    <cert-to-name>
      <id>2</id>
      <map-type>x509c2n:san-any</map-type>
    </cert-to-name>
  </client-identity-mappings>
</netconf-server-parameters>
</tls>

```

```

    </endpoint>
  <endpoint>
    <name>west-data-center</name>
    <tls>
      <tcp-client-parameters>
        <remote-address>west.analytics.example.com</remote-add\
ress>
        <keepalives>
          <idle-time>15</idle-time>
          <max-probes>3</max-probes>
          <probe-interval>30</probe-interval>
        </keepalives>
      </tcp-client-parameters>
      <tls-server-parameters>
        <server-identity>
          <certificate>
            <keystore-reference>
              <asymmetric-key>rsa-asymmetric-key</asymmetric-k\
ey>
              <certificate>ex-rsa-cert</certificate>
            </keystore-reference>
          </certificate>
        </server-identity>
        <client-authentication>
          <ca-certs>
            <truststore-reference>trusted-client-ca-certs</tru\
ststore-reference>
          </ca-certs>
          <ee-certs>
            <truststore-reference>trusted-client-ee-certs</tru\
ststore-reference>
          </ee-certs>
        </client-authentication>
      </tls-server-parameters>
      <netconf-server-parameters>
        <client-identity-mappings>
          <cert-to-name>
            <id>1</id>
            <fingerprint>11:0A:05:11:00</fingerprint>
            <map-type>x509c2n:specified</map-type>
            <name>scooby-doo</name>
          </cert-to-name>

```

```
        <cert-to-name>
          <id>2</id>
          <map-type>x509c2n:san-any</map-type>
        </cert-to-name>
      </client-identity-mappings>
    </netconf-server-parameters>
  </tls>
</endpoint>
</endpoints>
<connection-type>
  <persistent/>
</connection-type>
<reconnect-strategy>
  <start-with>first-listed</start-with>
  <max-attempts>3</max-attempts>
</reconnect-strategy>
</netconf-client>
</call-home>
</netconf-server>
```

### 3.3. YANG Module

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

<CODE BEGINS> file "ietf-netconf-server@2022-03-07.yang"

```
module ietf-netconf-server {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-netconf-server";
  prefix ncs;

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

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

  import ietf-tcp-client {
```

```
    prefix tcpc;
    reference
      "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
  }

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

  import ietf-ssh-common {
    prefix sshcmn;
    revision-date 2022-03-07; // stable grouping definitions
    reference
      "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
  }

  import ietf-ssh-server {
    prefix sshs;
    revision-date 2022-03-07; // stable grouping definitions
    reference
      "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
  }

  import ietf-tls-server {
    prefix tlss;
    revision-date 2022-03-07; // stable grouping definitions
    reference
      "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
  }

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

  contact
    "WG Web:  https://datatracker.ietf.org/wg/netconf
    WG List:  NETCONF WG list <mailto:netconf@ietf.org>
    Author:   Kent Watsen <mailto:kent+ietf@watsen.net>
    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) 2021 IETF Trust and the persons identified
```

as authors of the code. All rights reserved.

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

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

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

```
revision 2022-03-07 {
  description
    "Initial version";
  reference
    "RFC HHHH: NETCONF Client and Server Models";
}

// Features

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

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



```
}

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

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

// Groupings

grouping netconf-server-grouping {
  description
    "A reusable grouping for configuring a NETCONF server
    without any consideration for how underlying transport
    sessions are established.

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

  container client-identity-mappings {
    description
      "Specifies mappings through which NETCONF client X.509
      certificates are used to determine a NETCONF username,
      per RFC 7407.

      For TLS-based transports, 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, per Section 7 in RFC 7589."
```

```
For SSH-based transports, a matching cert-to-name
entry overrides the username provided by the SSH
implementation, consistent with the second paragraph
of Section 3 in RFC 6242.";
reference
  "RFC 6242:
    Using the NETCONF Protocol over Secure Shell (SSH)
  RFC 7589:
    Using the NETCONF Protocol over Transport Layer
    Security (TLS) with Mutual X.509 Authentication";
uses x509c2n:cert-to-name {
  refine "cert-to-name/fingerprint" {
    mandatory false;
    description
      "A 'fingerprint' value does not need to be specified
      when the 'cert-to-name' mapping is independent of
      fingerprint matching. A 'cert-to-name' having no
      fingerprint value will match any client certificate
      and therefore should only be present at the end of
      the user-ordered 'cert-to-name' list.";
  }
}
}
}

grouping netconf-server-listen-stack-grouping {
  description
    "A reusable grouping for configuring a NETCONF server
    'listen' protocol stack for a single connection.";
  choice transport {
    mandatory true;
    description
      "Selects between available transports.";
    case ssh {
      if-feature "ssh-listen";
      container ssh {
        description
          "SSH-specific listening configuration for inbound
          connections.";
        container tcp-server-parameters {
          description
            "A wrapper around the TCP client parameters
            to avoid name collisions.";
          uses tcps:tcp-server-grouping {
            refine "local-port" {
              default "830";
              description
                "The NETCONF server will listen on the
```

```
        IANA-assigned well-known port value
        for 'netconf-ssh' (830) if no value
        is specified.";
    }
}
container ssh-server-parameters {
  description
    "A wrapper around the SSH server parameters
    to avoid name collisions.";
  uses sshs:ssh-server-grouping;
}
container netconf-server-parameters {
  description
    "A wrapper around the NETCONF server parameters
    to avoid name collisions.";
  uses ncs:netconf-server-grouping {
    refine "client-identity-mappings" {
      if-feature "sshcmn:ssh-x509-certs";
      description
        "Augments in an 'if-feature' statement
        ensuring the 'client-identity-mappings'
        descendant is enabled only when SSH
        supports X.509 certificates.";
    }
    augment "client-identity-mappings" {
      description
        "Adds a flag indicating if a cert-to-name
        is required.";
      leaf mapping-required {
        type boolean;
        description
          "Indicates that the cert-to-name mapping
          is required (i.e., the SSH-level username
          is ignored).";
      }
    }
  }
}
}
}
case tls {
  if-feature "tls-listen";
  container tls {
    description
      "TLS-specific listening configuration for inbound
      connections.";
    container tcp-server-parameters {
```

```
description
  "A wrapper around the TCP client parameters
  to avoid name collisions.";
uses tcps:tcp-server-grouping {
  refine "local-port" {
    default "6513";
    description
      "The NETCONF server will listen on the
      IANA-assigned well-known port value
      for 'netconf-tls' (6513) if no value
      is specified.";
  }
}
}
container tls-server-parameters {
  description
    "A wrapper around the TLS server parameters to
    avoid name collisions.";
  uses tlss:tls-server-grouping {
    refine "client-authentication" {
      must 'ca-certs or ee-certs';
      description
        "NETCONF/TLS servers MUST validate client
        certificates. This configures certificates
        at the socket-level (i.e. bags), more
        discriminating client-certificate checks
        SHOULD be implemented by the application.";
      reference
        "RFC 7589:
        Using the NETCONF Protocol over Transport Layer
        Security (TLS) with Mutual X.509 Authentication";
    }
  }
}
}
container netconf-server-parameters {
  description
    "A wrapper around the NETCONF server parameters
    to avoid name collisions.";
  uses ncs:netconf-server-grouping {
    refine "client-identity-mappings/cert-to-name" {
      min-elements 1;
      description
        "The TLS transport requires a mapping.";
    }
  }
}
}
}
```

```
    }  
  }  
  
  grouping netconf-server-callhome-stack-grouping {  
    description  
      "A reusable grouping for configuring a NETCONF server  
      'call-home' protocol stack, for a single connection.";  
    choice transport {  
      mandatory true;  
      description  
        "Selects between available transports.";  
      case ssh {  
        if-feature "ssh-call-home";  
        container ssh {  
          description  
            "Specifies SSH-specific call-home transport  
            configuration.";  
          container tcp-client-parameters {  
            description  
              "A wrapper around the TCP client parameters  
              to avoid name collisions.";  
            uses tcpc:tcp-client-grouping {  
              refine "remote-port" {  
                default "4334";  
                description  
                  "The NETCONF server will attempt to connect  
                  to the IANA-assigned well-known port for  
                  'netconf-ch-tls' (4334) if no value is  
                  specified.";  
              }  
            }  
          }  
        }  
        container ssh-server-parameters {  
          description  
            "A wrapper around the SSH server parameters  
            to avoid name collisions.";  
          uses sshs:ssh-server-grouping;  
        }  
        container netconf-server-parameters {  
          description  
            "A wrapper around the NETCONF server parameters  
            to avoid name collisions.";  
          uses ncs:netconf-server-grouping {  
            refine "client-identity-mappings" {  
              if-feature "sshcmn:ssh-x509-certs";  
              description  
                "Augments in an 'if-feature' statement  
                ensuring the 'client-identity-mappings'
```

```
        descendant is enabled only when SSH
        supports X.509 certificates.";
    }
    augment "client-identity-mappings" {
        description
            "Adds a flag indicating if a cert-to-name
            is required.";
        leaf mapping-required {
            type boolean;
            description
                "Indicates that the cert-to-name mapping
                is required (i.e., the SSH-level username
                is ignored).";
        }
    }
}
}
}
}
}
}
case tls {
    if-feature "tls-call-home";
    container tls {
        description
            "Specifies TLS-specific call-home transport
            configuration.";
        container tcp-client-parameters {
            description
                "A wrapper around the TCP client parameters
                to avoid name collisions.";
            uses tcpc:tcp-client-grouping {
                refine "remote-port" {
                    default "4335";
                    description
                        "The NETCONF server will attempt to connect
                        to the IANA-assigned well-known port for
                        'netconf-ch-tls' (4335) if no value is
                        specified.";
                }
            }
        }
    }
    container tls-server-parameters {
        description
            "A wrapper around the TLS server parameters to
            avoid name collisions.";
        uses tlss:tls-server-grouping {
            refine "client-authentication" {
                must 'ca-certs or ee-certs';
                description

```

```
        "NETCONF/TLS servers MUST validate client
        certificates. This configures certificates
        at the socket-level (i.e. bags), more
        discriminating client-certificate checks
        SHOULD be implemented by the application.";
    reference
        "RFC 7589:
        Using the NETCONF Protocol over Transport Layer
        Security (TLS) with Mutual X.509 Authentication";
    }
}
}
container netconf-server-parameters {
    description
        "A wrapper around the NETCONF server parameters
        to avoid name collisions.";
    uses ncs:netconf-server-grouping {
        refine "client-identity-mappings/cert-to-name" {
            min-elements 1;
            description
                "The TLS transport requires a mapping.";
        }
    }
}
}
}
}
}
}

grouping netconf-server-app-grouping {
    description
        "A reusable grouping for configuring a NETCONF server
        application that supports both 'listen' and 'call-home'
        protocol stacks for a multiplicity of connections.";
    container listen {
        if-feature "ssh-listen or tls-listen";
        presence
            "Indicates that server-listening ports have been configured.
            This statement is present so the mandatory descendant
            nodes do not imply that this node must be configured.";
        description
            "Configures 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.";
        }
        uses netconf-server-listen-stack-grouping;
    }
}
container call-home {
    if-feature "ssh-call-home or tls-call-home";
    presence
        "Indicates that server-initiated call home connections have
        been configured. This statement is present so the mandatory
        descendant nodes do not imply that this node must be
        configured.";
    description
        "Configures the NETCONF server to initiate the underlying
        transport connection to NETCONF clients.";
    list netconf-client {
        key "name";
        min-elements 1;
        description
            "List of NETCONF clients the NETCONF server is to
            maintain simultaneous call-home connections with.";
        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.";
}
uses netconf-server-callhome-stack-grouping;
}
}
container connection-type {
  description
    "Indicates the NETCONF server's preference for how the
    NETCONF connection is maintained.";
  choice connection-type {
    mandatory true;
    description
      "Selects between available connection types.";
    case persistent-connection {
      container persistent {
        presence
          "Indicates that a persistent connection is to be
          maintained.";
        description
          "Maintain a persistent connection to the NETCONF
          client. If the connection goes down, immediately
          start trying to reconnect to the NETCONF client,
          using the reconnection strategy.

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

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

          The NETCONF client SHOULD gracefully close the
```

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

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

```

leaf period {
  type uint16;
  units "minutes";
  default "60";
  description
    "Duration of time between periodic connections.";
}
leaf anchor-time {
  type yang:date-and-time {
    // constrained to minute-level granularity
    pattern '\d{4}-\d{2}-\d{2}T\d{2}:\d{2}'
      + '(Z|[\+|-]\d{2}:\d{2})';
  }
  description
    "Designates a timestamp before or after which a
    series of periodic connections are determined.
    The periodic connections occur at a whole
    multiple interval from the anchor time. For
    example, for an anchor time is 15 minutes past
    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.";
}
} // case periodic-connection
} // choice connection-type
} // container connection-type

```

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

```
    } // list netconf-client
  } // container call-home
} // grouping netconf-server-app-grouping

// Protocol accessible node for servers that implement this module.
container netconf-server {
  uses netconf-server-app-grouping;
  description
    "Top-level container for NETCONF server configuration.";
}
}
```

<CODE ENDS>

#### 4. Security Considerations

##### 4.1. The "ietf-netconf-client" YANG Module

The "ietf-netconf-client" YANG module defines data nodes that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

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

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

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

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

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

#### 4.2. The "ietf-netconf-server" YANG Module

The "ietf-netconf-server" YANG module defines data nodes that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

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

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

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

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

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

### 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-netconf-client  
Registrant Contact: The IESG  
XML: N/A, the requested URI is an XML namespace.

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

## 5.2. The "YANG Module Names" Registry

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

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

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

## 6. References

### 6.1. Normative References

[I-D.ietf-netconf-keystore]

Watsen, K., "A YANG Data Model for a Keystore", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-23, 14 December 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-keystore-23>>.

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

Watsen, K., "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-ssh-client-server-26, 14 December 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-ssh-client-server-26>>.

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

Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tcp-client-server-11, 14 December 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tcp-client-server-11>>.

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

Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-26, 14 December 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tls-client-server-26>>.

- [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>>.

## 6.2. Informative References

- [I-D.ietf-netconf-crypto-types]  
Watsen, K., "YANG Data Types and Groupings for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-21, 14 September 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-crypto-types-21>>.
- [I-D.ietf-netconf-http-client-server]  
Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-08, 14 December 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-http-client-server-08>>.
- [I-D.ietf-netconf-netconf-client-server]  
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-24, 14 December 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-netconf-client-server-24>>.
- [I-D.ietf-netconf-restconf-client-server]  
Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-24, 14 December 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-restconf-client-server-24>>.
- [I-D.ietf-netconf-trust-anchors]  
Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-16, 14 December 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-trust-anchors-16>>.
- [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>>.
- [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

This section is to be removed before publishing as an RFC.

### 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 accommodate 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

## A.9. 08 to 09

- \* Corrected use of "mandatory true" for "address" leafs.
- \* Updated examples to reflect update to groupings defined in the keystore draft.

- \* Updated to use groupings defined in new TCP and HTTP drafts.
- \* Updated copyright date, boilerplate template, affiliation, and folding algorithm.

## A.10. 09 to 10

- \* Reformatted YANG modules.

## A.11. 10 to 11

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

## A.12. 11 to 12

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

## A.13. 12 to 13

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

## A.14. 13 to 14

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

## A.15. 14 to 15

- \* Refactored both the client and server modules similar to how the ietf-restconf-server module was refactored in -13 of that draft, and the ietf-restconf-client grouping.

## A.16. 15 to 16

- \* Added refinement to make "cert-to-name/fingerprint" be mandatory false.
- \* Commented out refinement to "tls-server-grouping/client-authentication" until a better "must" expression is defined.

## A.17. 16 to 17

- \* Updated examples to include the "\*-key-format" nodes.
- \* Updated examples to remove the "required" nodes.
- \* Updated examples to remove the "client-auth-defined-elsewhere" nodes.

## A.18. 17 to 18

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

## A.19. 18 to 19

- \* Updated examples to remove the 'algorithm' nodes.
- \* Updated examples to reflect the new TLS keepalives structure.
- \* Added keepalives to the tcp-client-parameters section in the netconf-server SSH-based call-home example.

- \* Added a TLS-based call-home example to the netconf-client example.
- \* Added a "Note to Reviewers" note to first page.

## A.20. 19 to 20

- \* Expanded "Data Model Overview section(s) [remove "wall" of tree diagrams].
- \* Removed expanded tree diagrams that were listed in the Appendix.
- \* Updated the Security Considerations section.

## A.21. 20 to 21

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

## A.22. 21 to 22

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

## A.23. 22 to 23

- \* Floated an 'if-feature' statement in a grouping down to where the grouping is used.
- \* Clarified 'client-identity-mappings' for both the SSH and TLS transports.
- \* For netconf-client, augmented-in a 'mapping-required' flag into 'client-identity-mappings' only for the SSH transport, and refined-in a 'min-elements 1' only for the TLS transport.
- \* Aligned modules with 'pyang -f' formatting.

## A.24. 23 to 24

- \* Replaced "base64encodedvalue==" with "BASE64VALUE=" in examples.
- \* Minor editorial nits

## A.25. 24 to 25

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

#### Acknowledgements

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

#### Author's Address

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

NETCONF  
Internet-Draft  
Intended status: Standards Track  
Expires: May 4, 2017

A. Gonzalez Prieto  
Cisco Systems  
A. Clemm  
Sympotech  
E. Voit  
E. Nilsen-Nygaard  
A. Tripathy  
Cisco Systems  
S. Chisholm  
Ciena  
H. Trevino  
Cisco Systems  
October 31, 2016

NETCONF Support for Event Notifications  
draft-ietf-netconf-netconf-event-notifications-01

Abstract

This document defines the support of [event-notifications] by the Network Configuration protocol (NETCONF). [event-notifications] describes capabilities and operations for providing asynchronous message notification delivery. This document discusses how to provide them on top of NETCONF. The capabilities and operations defined between this document and [event-notifications] are intended to obsolete RFC 5277.

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 <http://datatracker.ietf.org/drafts/current/>.

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

This Internet-Draft will expire on May 4, 2017.

## Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

1. Introduction . . . . .	3
1.1. Terminology . . . . .	4
1.2. Solution Overview . . . . .	5
2. Solution . . . . .	5
2.1. Event Streams . . . . .	6
2.2. Event Stream Discovery . . . . .	6
2.3. Default Event Stream . . . . .	9
2.4. Creating a Subscription . . . . .	9
2.5. Establishing a Subscription . . . . .	11
2.6. Modifying a Subscription . . . . .	16
2.7. Deleting a Subscription . . . . .	21
2.8. Configured Subscriptions . . . . .	24
2.9. Event (Data Plane) Notifications . . . . .	33
2.10. Control Plane Notifications . . . . .	35
3. Backwards Compatibility . . . . .	44
3.1. Capabilities . . . . .	44
3.2. Stream Discovery . . . . .	45
4. Security Considerations . . . . .	45
5. Acknowledgments . . . . .	46
6. References . . . . .	46
6.1. Normative References . . . . .	46
6.2. Informative References . . . . .	47
Appendix A. Issues that are currently being worked . . . . .	47
Appendix B. Changes between revisions . . . . .	47
B.1. v00 to v01 . . . . .	47
Authors' Addresses . . . . .	47



## 1. Introduction

[RFC6241] can be conceptually partitioned into four layers:

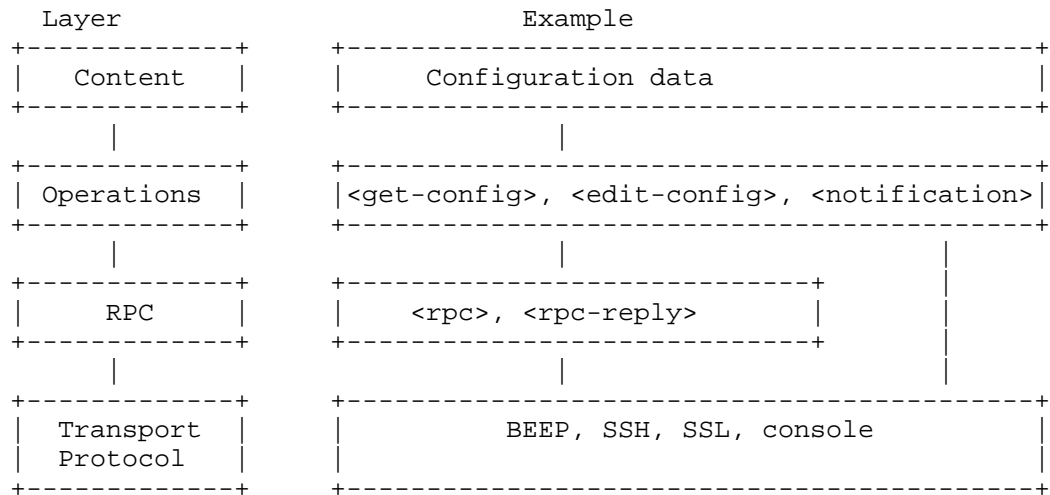


Figure 1: NETCONF layer architecture

This document defines mechanisms that provide an asynchronous message notification delivery service for the NETCONF protocol [RFC6241] based on [event-notifications]. This is an optional capability built on top of the base NETCONF definition.

[event-notifications] and this document enhance the capabilities of RFC 5277 while maintaining backwards capability with existing implementations. It is intended that a final version of this document might obsolete [RFC5277]. The enhancements include the ability to terminate subscriptions without terminating the client session, to modify existing subscriptions, and to have multiple subscriptions on a NETCONF session. [RFC5277] clients that do not require these enhancements are not affected by them.

[event-notifications] covers the following functionality:

- o Ability to subscribe to event notifications using two mechanisms: dynamic and configuration subscriptions.
- o Ability to subscribe to event notifications using two mechanisms: dynamic and configuration subscriptions.
- o Ability to negotiate acceptable subscription parameters.

- o Ability to filter the subset of notifications to be pushed with stream-specific semantics.
- o Ability to support multiple encodings for the notification.
- o Mechanism to communicate the notifications.
- o Ability to replay locally logged notifications.

To support this functionality, NETCONF agents must implement the operations, configuration and operational state defined in [event-notifications]. In addition, they need to:

- o support multiple subscriptions over a single NETCONF session.
- o support a revised definition of the default NETCONF stream
- o be backwards compatible with RFC 5277 implementations.

#### 1.1. 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].

##### 1.1.1. NETCONF

The following terms are defined in [RFC6241] :

- o Client
- o Server
- o Operation
- o RPC: remote procedure call

##### 1.1.2. Event Notifications

The following terms are defined in [event-notifications]:

- o Event
- o Event notification
- o Stream (also referred to as "event stream")
- o Subscriber

- o Publisher
- o Receiver
- o Subscription
- o Filter
- o Dynamic subscription
- o Configured subscription

Note that a publisher in [event-notifications] corresponds to a server in [RFC6241]. Similarly, a subscribers corresponds to a client. A receiver is also a client. In the remainder of this document, we will use the terminology in [RFC6241].

#### 1.1.3. NETCONF Access Control

The following terms are defined in [RFC6536]:

- o NACM: NETCONF Access Control Model

#### 1.2. Solution Overview

[event-notifications] defines mechanisms that provide an asynchronous message notification delivery service. This document discusses its realization on top of the NETCONF protocol [RFC6241].

The functionality to support is defined in [event-notifications]. It is formalized in a set of yang models. The mapping of yang constructs into NETCONF is described in [RFC6020].

Supporting [event-notifications] requires enhancements and modifications in NETCONF. The key enhancement is supporting multiple subscriptions on a NETCONF session. A key modification is the definition of the NETCONF stream.

These enhancements do not affect [RFC5277] clients that do not support [event-notifications].

## 2. Solution

In this section, we describe and exemplify how [event-notifications] must be supported over NETCONF.

## 2.1. Event Streams

In the context of NETCONF, an event stream is a set of events available for subscription from a NETCONF server. It is out of the scope of this document to identify a) how streams are defined, b) how events are defined/generated, and c) how events are assigned to streams.

The following is a high-level description of the flow of a notification. Note that it does not mandate and/or preclude an implementation. As events are raised, they are assigned to streams. An event may be assigned to multiple streams. The event is distributed to subscribers and receivers based on the current subscriptions and access control. Access control is needed because if any receiver of that subscription does not have permission to receive an event, then it never makes it into a notification, and processing of the event is completed for that subscription.

## 2.2. Event Stream Discovery

A NETCONF client can retrieve the list of available event streams from a NETCONF server using the <get> operation. The reply contains the elements defined in the YANG model under the container "/streams", which includes the stream identifier.

The following example illustrates the retrieval of the list of available event streams using the <get> operation.

```
<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-event-notifications"/>
      </filter>
    </get>
  </rpc>
```

Figure 2: Get streams

The NETCONF server returns a list of event streams available for subscription. In this example, the list contains the NETCONF, SNMP, and syslog-critical streams.

```
<rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <streams
      xmlns="urn:ietf:params:xml:ns:yang:ietf-event-notifications">
      <stream>NETCONF</stream>
      <stream>SNMP</stream>
      <stream>syslog-critical</stream>
      <stream>NETCONF</stream>
    </streams>
  </data>
</rpc-reply>
```

Figure 3: Get streams response

### 2.2.1. Backwards Compatibility

In order to maintain backwards compatibility, clients that only support [RFC5277] can retrieve the list of available event streams executing a <get> operation against the container "/netconf/streams".

The following example illustrates this mechanism.

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

Figure 4: Get streams (backwards compatibility)

The NETCONF server returns a list of event streams available for subscription. In this example, the list contains the NETCONF, SNMP, and syslog-critical streams.

```
<rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <netconf
      xmlns="urn:ietf:params:xml:ns:netmod:notification">
      <streams>
        <stream>
```

```
<name>
  NETCONF
</name>
<description>
  default NETCONF event stream
</description>
<replaySupport>
  true
</replaySupport>
<replayLogCreationTime>
  2016-02-05T00:00:00Z
</replayLogCreationTime>
</stream>
<stream>
  <name>
    SNMP
  </name>
  <description>
    SNMP notifications
  </description>
  <replaySupport>
    false
  </replaySupport>
</stream>
<stream>
  <name>
    syslog-critical
  </name>
  <description>
    Critical and higher severity
  </description>
  <replaySupport>
    true
  </replaySupport>
  <replayLogCreationTime>
    2007-07-01T00:00:00Z
  </replayLogCreationTime>
</stream>
</streams>
</netconf>
</data>
</rpc-reply>
```

Figure 5: Get streams response (backwards compatibility)

### 2.3. Default Event Stream

A NETCONF server implementation supporting the notification capability MUST support the "NETCONF" notification event stream. This stream contains all NETCONF XML event notifications supported by the NETCONF server, except for those belonging only to streams that explicitly indicate that they must be excluded from the NETCONF stream. The exact string "NETCONF" is used during the advertisement of stream support during the <get> operation on <streams> and during the <create-subscription> and <establish-subscription> operations.

### 2.4. Creating a Subscription

This operation was fully defined in [RFC5277].

#### 2.4.1. Usage Example

The following demonstrates dynamically creating a subscription.

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

Figure 6: Create subscription

#### 2.4.2. Positive Response

If the NETCONF server can satisfy the request, the server sends an <ok> element.

```
<netconf:rpc netconf:message-id="102"
  xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
  <create-subscription
    xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
    <filter netconf:type="xpath"
      xmlns:ex="http://example.com/event/1.0"
      select="/ex:event[ex:eventClass='fault' and
        (ex:severity='minor' or ex:severity='major'
        or ex:severity='critical')]" />
    </create-subscription>
  </netconf:rpc>

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

Figure 7: Successful create subscription

#### 2.4.3. Negative Response

If the request cannot be completed for any reason, an `<rpc-error>` element is included within the `<rpc-reply>`. Subscription requests can fail for several reasons including if a filter with invalid syntax is provided or if the name of a non-existent stream is provided.

If a `stopTime` is specified in a request without having specified a `startTime`, the following error is returned:

```
Tag: missing-element
Error-type: protocol
Severity: error
Error-info: <bad-element>: startTime
Description: An expected element is missing.
```

Figure 8: Create subscription missing an element

If the optional replay feature is requested but the NETCONF server does not support it, the following error is returned:



Tag: operation-failed  
Error-type: protocol  
Severity: error  
Error-info: none  
Description: Request could not be completed because the  
                  requested operation failed for some reason  
                  not covered by any other error condition.

Figure 9: Create subscription operation failed

If a stopTime is requested that is earlier than the specified  
startTime, the following error is returned:

Tag: bad-element  
Error-type: protocol  
Severity: error  
Error-info: <bad-element>: stopTime  
Description: An element value is not correct;  
                  e.g., wrong type, out of range, pattern mismatch.

Figure 10: Create subscription incorrect stopTime

If a startTime is requested that is later than the current time, the  
following error is returned:

Tag: bad-element  
Error-type: protocol  
Severity: error  
Error-info: <bad-element>: startTime  
Description: An element value is not correct;  
                  e.g., wrong type, out of range, pattern mismatch.

Figure 11: Create subscription incorrect startTime

## 2.5. Establishing a Subscription

This operation is defined in [event-notifications].

### 2.5.1. Usage Example

The following illustrates the establishment of a simple subscription.

```

<netconf:rpc message-id="101"
  xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
  <establish-subscription
    xmlns="urn:ietf:params:xml:ns:netconf:notification:1.1">
  </establish-subscription>
</netconf:rpc>

```

Figure 12: Establish subscription

### 2.5.2. Positive Response

If the NETCONF server can satisfy the request, the server sends a positive <subscription-result> element, and the subscription-id of the accepted subscription.

```

<netconf:rpc netconf:message-id="102"
  xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
  <establish-subscription
    xmlns="urn:ietf:params:xml:ns:netconf:notification:1.1">
    <filter netconf:type="xpath"
      xmlns:ex="http://example.com/event/1.0"
      select="/ex:event[ex:eventClass='fault' and
        (ex:severity='minor' or ex:severity='major'
        or ex:severity='critical')]" />
    </establish-subscription>
  </netconf:rpc>

<rpc-reply message-id="102"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <subscription-result
    xmlns="urn:ietf:params:xml:ns:netconf:notification:1.1">
    ok
  </subscription-result>
  <subscription-id
    xmlns="urn:ietf:params:xml:ns:netconf:notification:1.1">
    52
  </subscription-id>
</rpc-reply>

```

Figure 13: Successful establish-subscription

### 2.5.3. Negative Response

If the NETCONF server cannot satisfy the request, the server sends a negative <subscription-result> element.

If the client has no authorization to establish the subscription, the `<subscription-result>` indicates an authorization error. For instance:

```
<netconf:rpc netconf:message-id="103"
  xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
  <establish-subscription
    xmlns="urn:ietf:params:xml:ns:netconf:notification:1.1">
    <stream>foo</stream>
  </establish-subscription>
</netconf:rpc>

<rpc-reply message-id="103"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <subscription-result
    xmlns="urn:ietf:params:xml:ns:netconf:notification:1.1">
    error-data-not-authorized
  </subscription-result>
</rpc-reply>
```

Figure 14: Unsuccessful establish subscription

If the request is rejected because the server is not able to serve it, the server SHOULD include in the returned error what subscription parameters would have been accepted for the request when it was processed. However, there is no guarantee that subsequent requests with those parameters for this client or others will be accepted. For instance, consider a subscription from [yang-push], which augments the `establish-subscription` with some additional parameters, including "period". If the client requests a period the NETCONF server cannot serve, the exchange may be:

```
<netconf:rpc message-id="101"
  xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
  <establish-subscription
    xmlns="urn:ietf:params:xml:ns:netconf:notification:1.1">
    <stream>push-update</stream>
    <filter netconf:type="xpath"
      xmlns:ex="http://example.com/sample-data/1.0"
      select="/ex:foo"/>
    <period xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
      500
    </period>
    <encoding>encode-xml</encoding>
  </establish-subscription>
</netconf:rpc>

<rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <subscription-result
    xmlns="urn:ietf:params:xml:ns:netconf:notification:1.1">
    error-insufficient-resources
  </subscription-result>
  <period
    xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
    2000
  </period>
</rpc-reply>
```

Figure 15: Subscription establishment negotiation

Subscription requests will fail if a filter with invalid syntax is provided or if the name of a non-existent stream is provided.

#### 2.5.4. Multiple Subscriptions over a Single NETCONF Session

Note that [event-notifications] requires supporting multiple subscription establishments over a single NETCONF session. In contrast, [RFC5277] mandated servers to return an error when a create-subscription was sent while a subscription was active on that session. Note that servers are not required to support multiple create-subscription over a single session, but they MUST support multiple establish-subscription over one session.

#### 2.5.5. Message Flow Examples

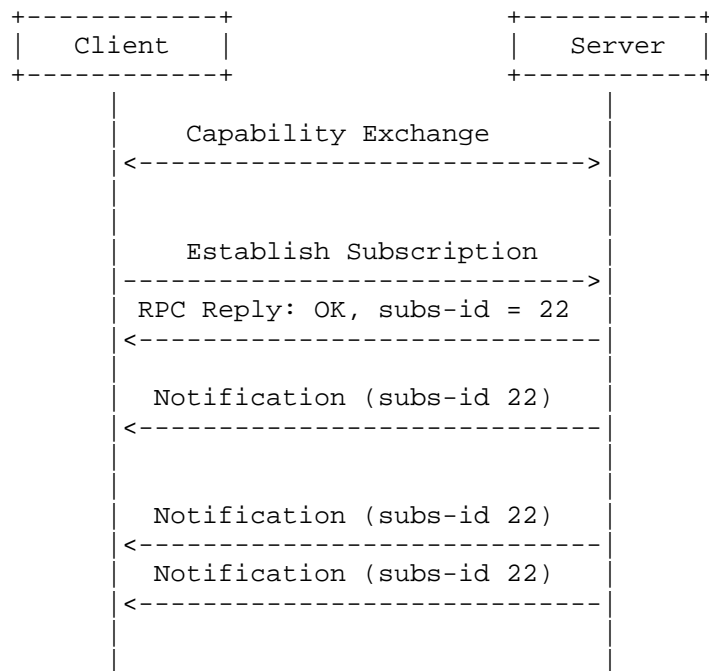


Figure 16: Message flow for subscription establishment

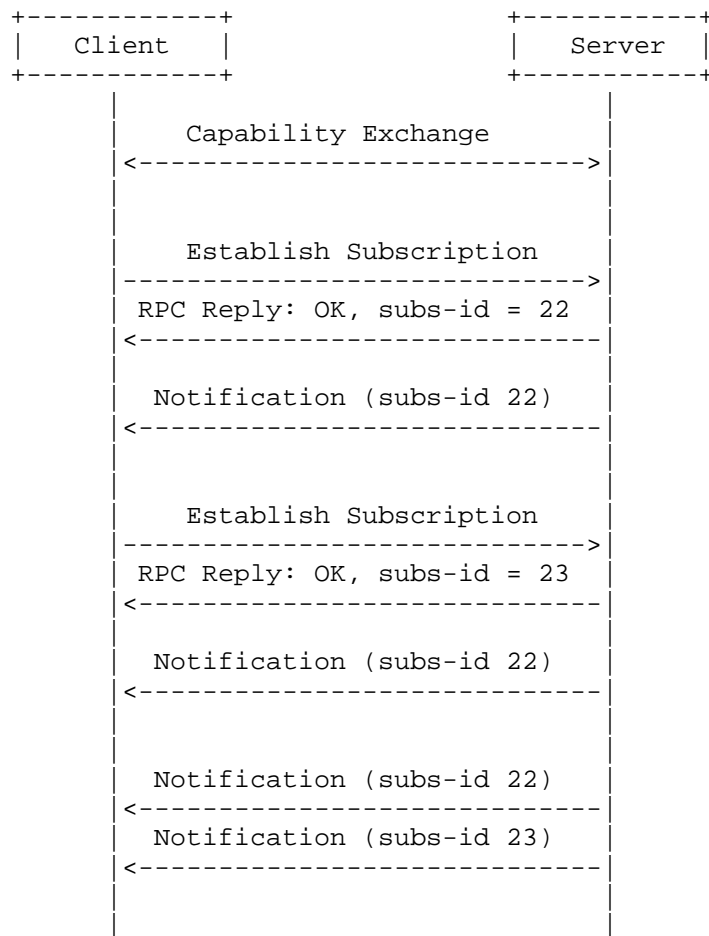


Figure 17: Message Flow for multiple subscription establishments over a single session

## 2.6. Modifying a Subscription

This operation is defined in [event-notifications].

### 2.6.1. Usage Example

The following demonstrates modifying a subscription. Consider a subscription from [yang-push], which augments the establish-subscription with some additional parameters, including "period". A subscription may be established as follows.

```
<netconf:rpc message-id="101"
  xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
  <establish-subscription
    xmlns="urn:ietf:params:xml:ns:netconf:notification:1.1">
    <stream>push-update</stream>
    <filter netconf:type="xpath"
      xmlns:ex="http://example.com/sample-data/1.0"
      select="/ex:foo"/>
    <period xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
      500
    </period>
    <encoding>encode-xml</encoding>
  </establish-subscription>
</netconf:rpc>

<rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <subscription-result
    xmlns="urn:ietf:params:xml:ns:netconf:notification:1.1">
    ok
  </subscription-result>
  <subscription-id
    xmlns="urn:ietf:params:xml:ns:netconf:notification:1.1">
    1922
  </subscription-id>
</rpc-reply>
```

Figure 18: Establish subscription to be modified

The subscription may be modified with:

```
<netconf:rpc message-id="102"
  xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
  <modify-subscription
    xmlns="urn:ietf:params:xml:ns:netconf:notification:1.1">
    <subscription-id>1922</subscription-id>
    <period>1000</period>
  </modify-subscription >
</netconf:rpc>

<rpc-reply message-id="102"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <subscription-result
    xmlns="urn:ietf:params:xml:ns:netconf:notification:1.1">
    ok
  </subscription-result>
  <subscription-id
    xmlns="urn:ietf:params:xml:ns:netconf:notification:1.1">
    1922
  </subscription-id>
</rpc-reply>
```

Figure 19: Modify subscription

#### 2.6.2. Positive Response

If the NETCONF server can satisfy the request, the server sends a positive `<subscription-result>` element. This response is like that to an establish-subscription request, but without the subscription-id (which would be redundant).



```
<netconf:rpc message-id="102"
  xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
  <modify-subscription
    xmlns="urn:ietf:params:xml:ns:netconf:notification:1.1">
    <subscription-id>1922</subscription-id>
    <period
      xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
      1000
    </period>
  </modify-subscription >
</netconf:rpc>

<rpc-reply message-id="102"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <subscription-result
    xmlns="urn:ietf:params:xml:ns:netconf:notification:1.1">
    ok
  </subscription-result>
</rpc-reply>
```

Figure 20: Successful modify subscription

### 2.6.3. Negative Response

If the NETCONF server cannot satisfy the request, the server sends a negative `<subscription-result>` element. Its contents and semantics are identical to those in an establish-subscription request. For instance:

```
<netconf:rpc message-id="102"
  xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
  <modify-subscription
    xmlns="urn:ietf:params:xml:ns:netconf:notification:1.1">
    <subscription-id>1922</subscription-id>
    <period xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
      100
    </period>
  </modify-subscription>
</netconf:rpc>

<rpc-reply message-id="102"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <subscription-result
    xmlns="urn:ietf:params:xml:ns:netconf:notification:1.1">
    error-insufficient-resources
  </subscription-result>
  <period>500</period>
</rpc-reply>
```

Figure 21: Unsuccessful modify subscription

#### 2.6.4. Message Flow Example

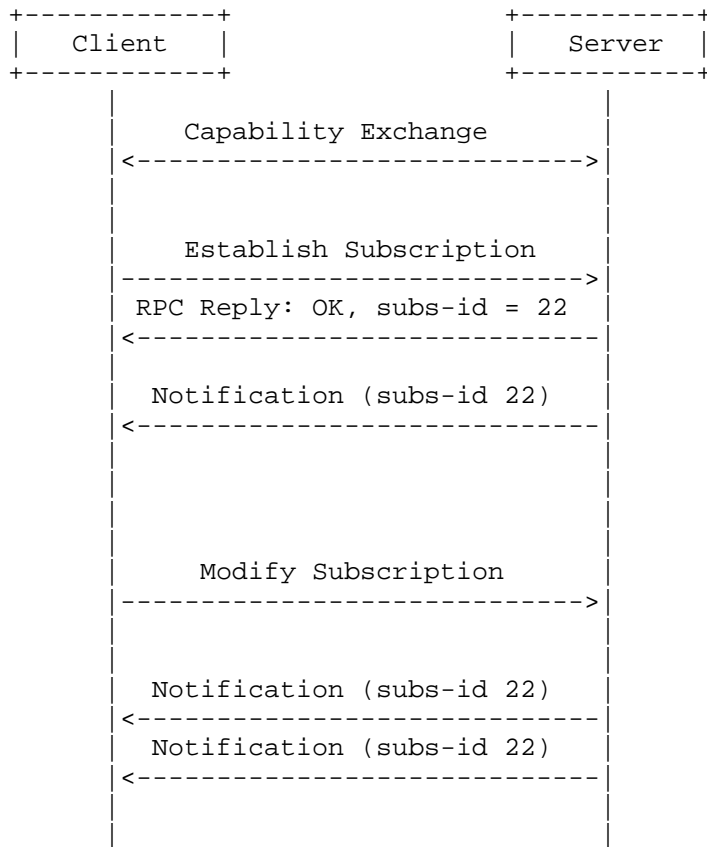


Figure 22: Message flow for subscription modification

## 2.7. Deleting a Subscription

This operation is defined in [event-notifications].

### 2.7.1. Usage Example

The following demonstrates deleting a subscription.

```
<netconf:rpc message-id="101"
  xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
  <delete-subscription
    xmlns="urn:ietf:params:xml:ns:netconf:notification:1.1">
    <subscription-id>1922</subscription-id>
  </delete-subscription>
</netconf:rpc>
```

Figure 23: Delete subscription

#### 2.7.2. Positive Response

If the NETCONF server can satisfy the request, the server sends an OK element. For example:

```
<netconf:rpc message-id="103"
  xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
  <delete-subscription
    xmlns="urn:ietf:params:xml:ns:netconf:notification:1.1">
    <subscription-id>1922</subscription-id>
  </delete-subscription>
</netconf:rpc>

<rpc-reply message-id="103"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

Figure 24: Successful delete subscription

#### 2.7.3. Negative Response

If the NETCONF server cannot satisfy the request, the server sends an error-rpc element. For example:

```
<netconf:rpc message-id="103"
  xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
  <delete-subscription
    xmlns="urn:ietf:params:xml:ns:netconf:notification:1.1">
    <subscription-id>2017</subscription-id>
  </delete-subscription>
</netconf:rpc>

<rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <rpc-error>
    <error-type>application</error-type>
    <error-tag>invalid-value</error-tag>
    <error-severity>error</error-severity>
    <error-path
      xmlns:t="urn:ietf:params:xml:ns:netconf:notification:1.1">
      /t:subscription-id
    </error-path>
    <error-message xml:lang="en">
      Subscription-id 2017 does not exist
    </error-message>
  </rpc-error>
</rpc-reply>
```

Figure 25: Unsuccessful delete subscription

#### 2.7.4. Message Flow Example

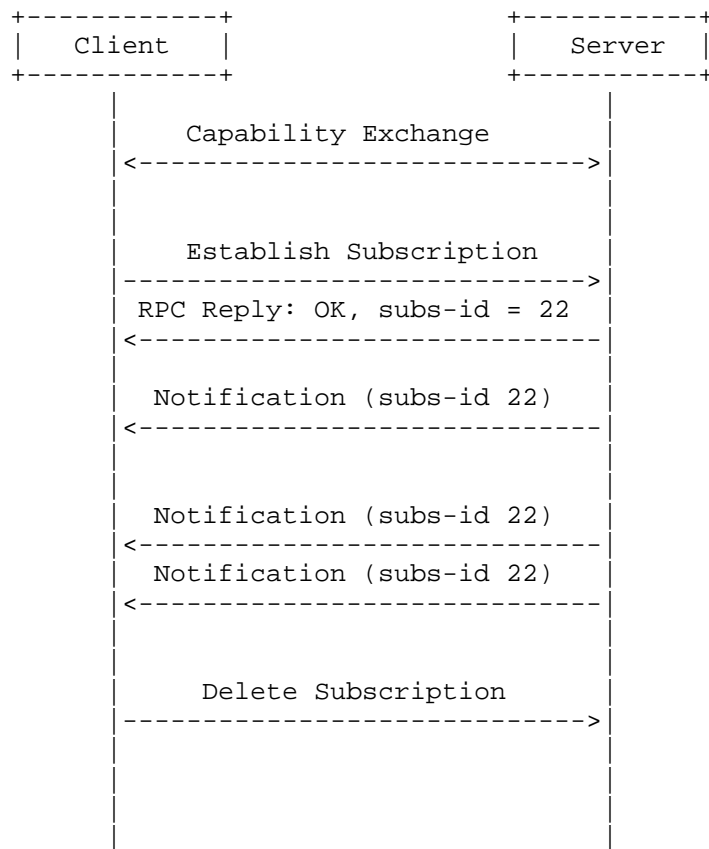


Figure 26: Message flow for subscription deletion

## 2.8. Configured Subscriptions

A configured subscription is a subscription installed via a configuration interface. Configured subscriptions do not support negotiation.

Supporting configured subscriptions is optional and advertised during the capabilities exchange using the "configured-subscriptions" feature.

Configured subscriptions are supported by NETCONF servers using NETCONF Call Home [call-home]

In this section, we present examples of how to manage configuration subscriptions using a NETCONF client.

### 2.8.1. Call Home for Configured Subscriptions

Configured subscriptions are established, modified, and deleted using configuration operations against the top-level subtree subscription-config. Once the configuration is set, the server initiates a Call Home to each of the receivers in the subscription on the address and port specified. Once the NETCONF session between the server and the receiver is established, the server will issue a "subscription-started" notification. After that, the server will send notifications to the receiver as per the subscription notification.

Note that the server assumes the receiver is aware that calls on the configured port are intended only for pushing notifications. It also assumes that the receiver is ready to accept notifications on the session created as part of the Call Home as soon as the NETCONF session is established. This may require coordination between the client that configures the subscription and the clients for which the notifications are intended. This coordination is out of the scope of this document.

### 2.8.2. Establishing a Configured Subscription

Subscriptions are established using configuration operations against the top-level subtree subscription-config.

For example at subscription establishment, a NETCONF client may send:

```
<rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
  xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <subscription-config
      xmlns="urn:ietf:params:xml:ns:netconf:notification:1.1">
      <subscription>
        <subscription-id>
          1922
        </subscription-id>
        <stream>
          foo
        </stream>
        <receiver>
          <address>
            1.2.3.4
          </address>
          <port>
            1234
          </port>
        </receiver>
      </subscription>
    </subscription-config>
  </edit-config>
</rpc>
```

Figure 27: Establish static subscription

if the request is accepted, the server would reply:

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

Figure 28: Response to a successful static subscription establishment

if the request is not accepted because the server cannot serve it,  
the server may reply:



```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <rpc-error>
    <error-type>application</error-type>
    <error-tag>resource-denied</error-tag>
    <error-severity>error</error-severity>
    <error-message xml:lang="en">
      Temporarily the server cannot serve this
      subscription due to the current workload.
    </error-message>
  </rpc-error>
</rpc-reply>
```

Figure 29: Response to a failed static subscription establishment

#### 2.8.2.1. Message Flow Example

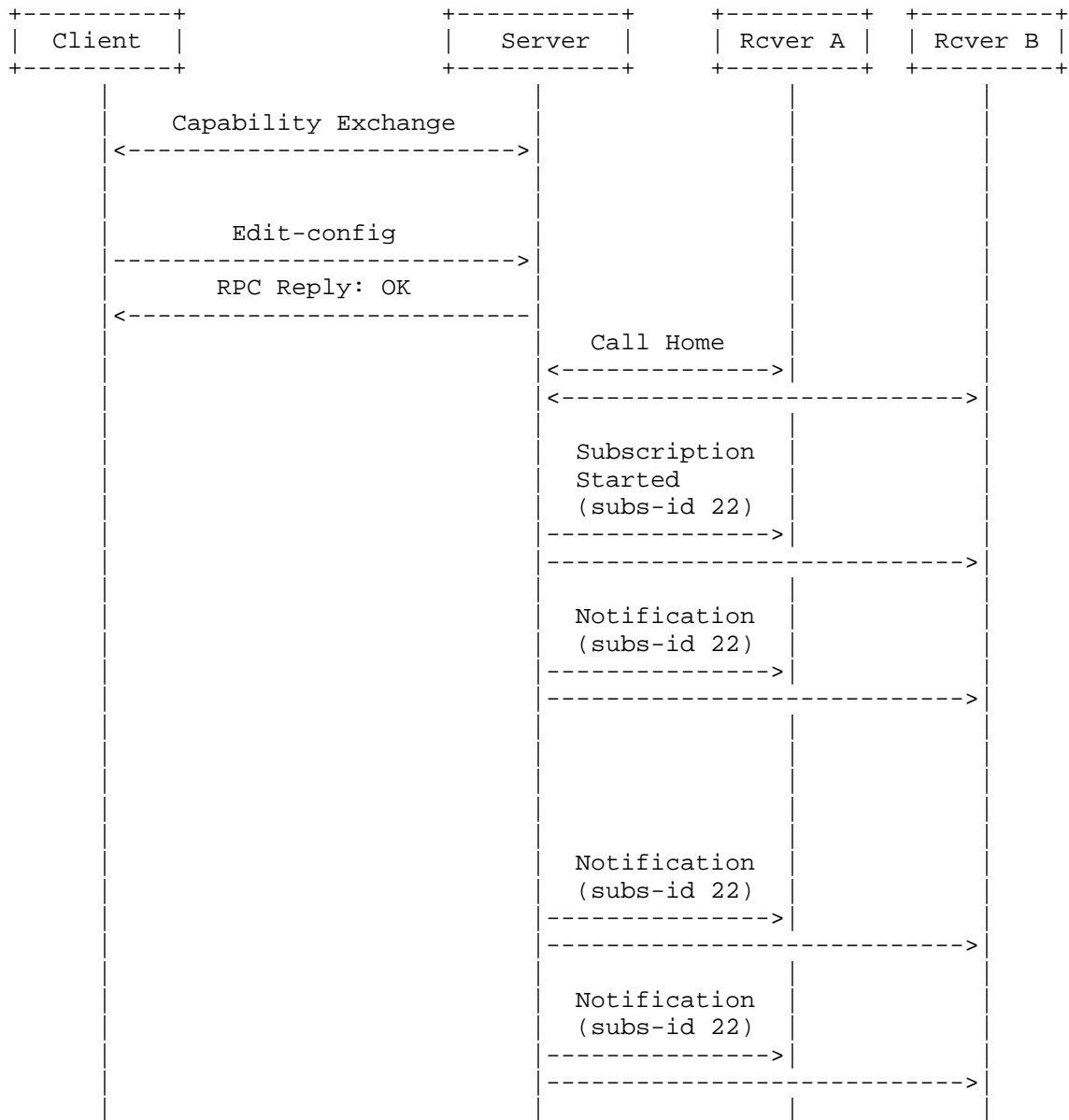


Figure 30: Message flow for subscription establishment (configured subscription)

### 2.8.3. Modifying a Configured Subscription

Configured subscriptions can be modified using configuration operations against the top-level subtree subscription-config.

For example, the subscription established in the previous section could be modified as follows, choosing a different receiver:

```
<rpc message-id="102"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
  xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <subscription-config
      xmlns="urn:ietf:params:xml:ns:netconf:notification:1.1">
      <subscription>
        <subscription-id>
          1922
        </subscription-id>
        <stream>
          foo
        </stream>
        <receiver>
          <address>
            1.2.3.5
          </address>
          <port>
            1234
          </port>
        </receiver>
      </subscription>
    </subscription-config>
  </edit-config>
</rpc>
```

Figure 31: Modify configured subscription

if the request is accepted, the server would reply:

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

Figure 32: Response to a successful configured subscription modification

## 2.8.3.1. Message Flow Example

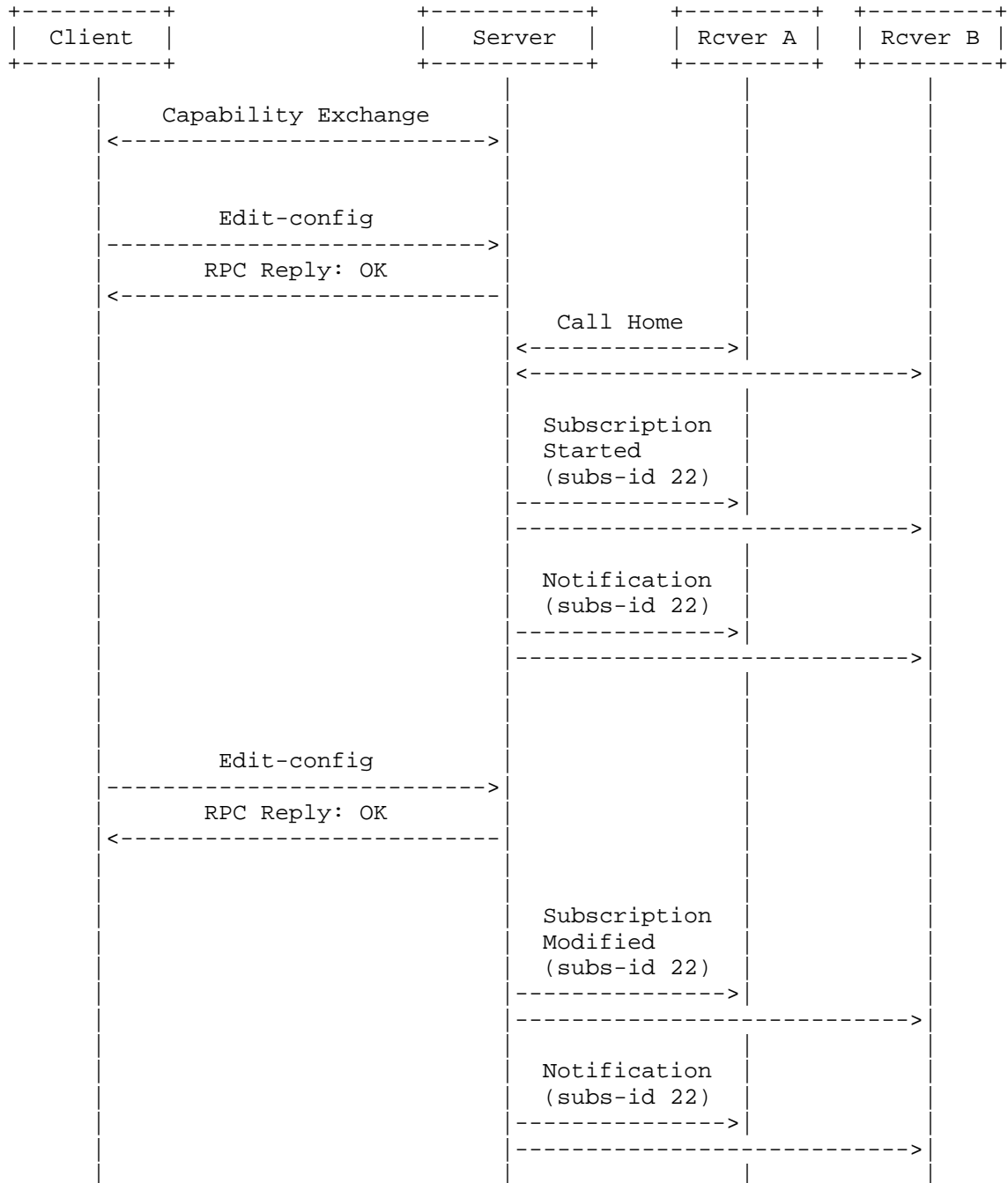


Figure 33: Message flow for subscription modification (configured subscription)

#### 2.8.4. Deleting a Configured Subscription

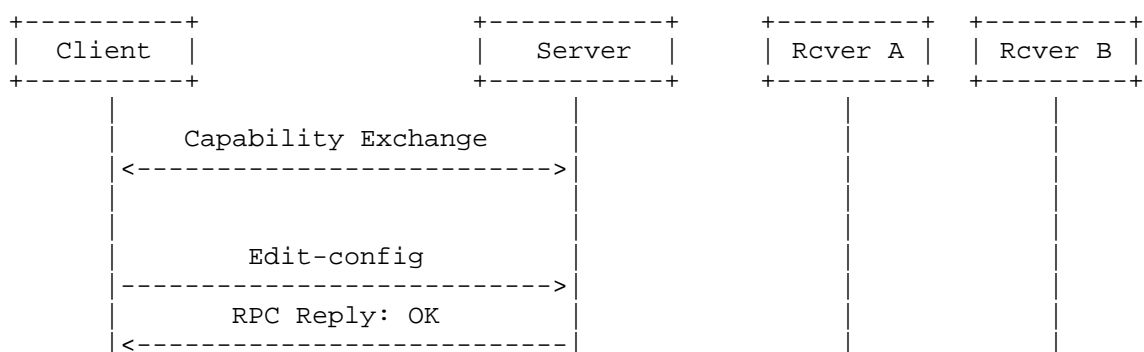
Subscriptions can be deleted using configuration operations against the top-level subtree subscription-config. For example:

```
<rpc message-id="103"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <subscription-config
      xmlns:xc="urn:ietf:params:xml:ns:netconf:notification:1.1">
      <subscription xc:operation="delete">
        <subscription-id>
          1922
        </subscription-id>
      </subscription>
    </subscription-config>
  </edit-config>
</rpc>

<rpc-reply message-id="103"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

Figure 34: Deleting a configured subscription

##### 2.8.4.1. Message Flow Example



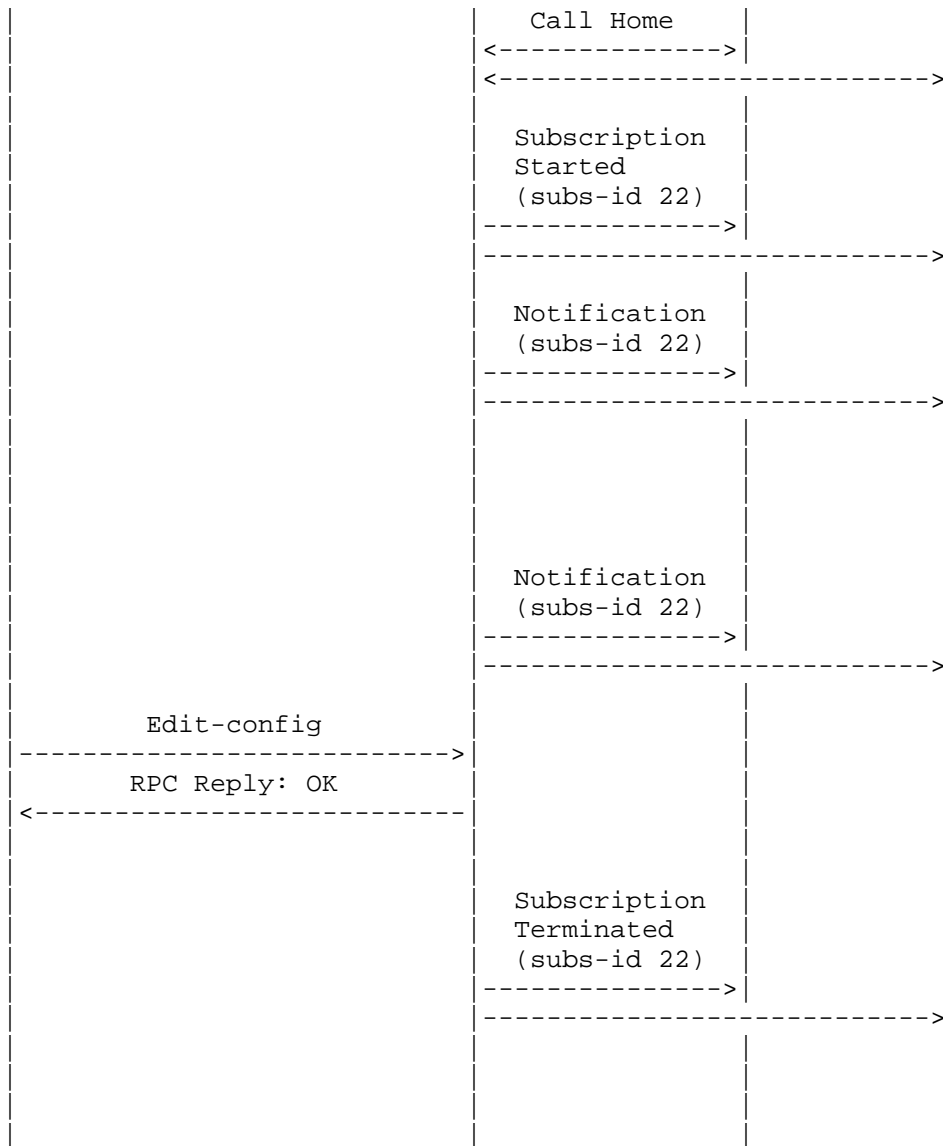


Figure 35: Message flow for subscription deletion (configured subscription)

## 2.9. Event (Data Plane) Notifications

Once a subscription has been set up, the NETCONF server sends (asynchronously) the event notifications from the subscribed stream. We refer to these as data plane notifications. For dynamic subscriptions set up via RPC operations, event notifications are sent over the NETCONF session used to create or establish the subscription. For static subscriptions, event notifications are sent over the specified connections.

An event notification is sent to the receiver(s) when an event of interest (i.e., meeting the specified filtering criteria) has occurred. An event notification is a complete and well-formed XML document. Note that `<notification>` is not a Remote Procedure Call (RPC) method but rather the top-level element identifying the one-way message as a notification. Note that event notifications never trigger responses.

The event notification always includes an `<eventTime>` element. It is the time the event was generated by the event source. This parameter is of type `dateTime` and compliant to [RFC3339]. Implementations must support time zones.

The event notification also contains notification-specific tagged content, if any. With the exception of `<eventTime>`, the content of the notification is beyond the scope of this document.

For encodings other than XML, notifications include an additional XML element so that the notification is a well-formed XML. The element is `<notification-contents-{encoding}>`, E.g., `<notification-contents-json>`. That element contains the notification contents in the desired encoding

The following is an example of an event notification from [RFC6020]:

```

notification link-failure {
  description "A link failure has been detected";
  leaf if-name {
    type leafref {
      path "/interface/name";
    }
  }
  leaf if-admin-status {
    type admin-status;
  }
  leaf if-oper-status {
    type oper-status;
  }
}

```

Figure 36: Definition of a data plane notification

```

<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 37: Data plane notification

The equivalent using JSON encoding would be

```

<notification
  xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2007-09-01T10:00:00Z</eventTime>
  <notification-contents-json>
    {
      "acme-system:link-failure": {
        "if-name": "so-1/2/3.0",
        "if-admin-status": "up",
        "if-oper-status": "down"
      }
    }
  </notification-contents-json>
</notification>

```

Figure 38: Data plane notification using JSON encoding



## 2.10. Control Plane Notifications

In addition to data plane notifications, a server may send control plane notifications (defined in [event-notifications]) to indicate to receivers that an event related to the subscription management has occurred. Control plane notifications cannot be filtered out. Next we exemplify them using both XML, and JSON encodings for the notification-specific content:

### 2.10.1. replayComplete

```
<notification
  xmlns=" urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2007-09-01T10:00:00Z</eventTime>
  <replayComplete xmlns="urn:ietf:params:xml:ns:netmod:notification"/>
</notification>
```

Figure 39: replayComplete control plane notification

The equivalent using JSON encoding would be:

```
<notification
  xmlns=" urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2007-09-01T10:00:00Z</eventTime>
  <notification-contents-json>
  {
    "netmod-notif:replayComplete": { }
  }
  </notification-contents-json>
</notification>
```

Figure 40: replayComplete control plane notification (JSON encoding)

#### 2.10.1.1. Message Flow Example

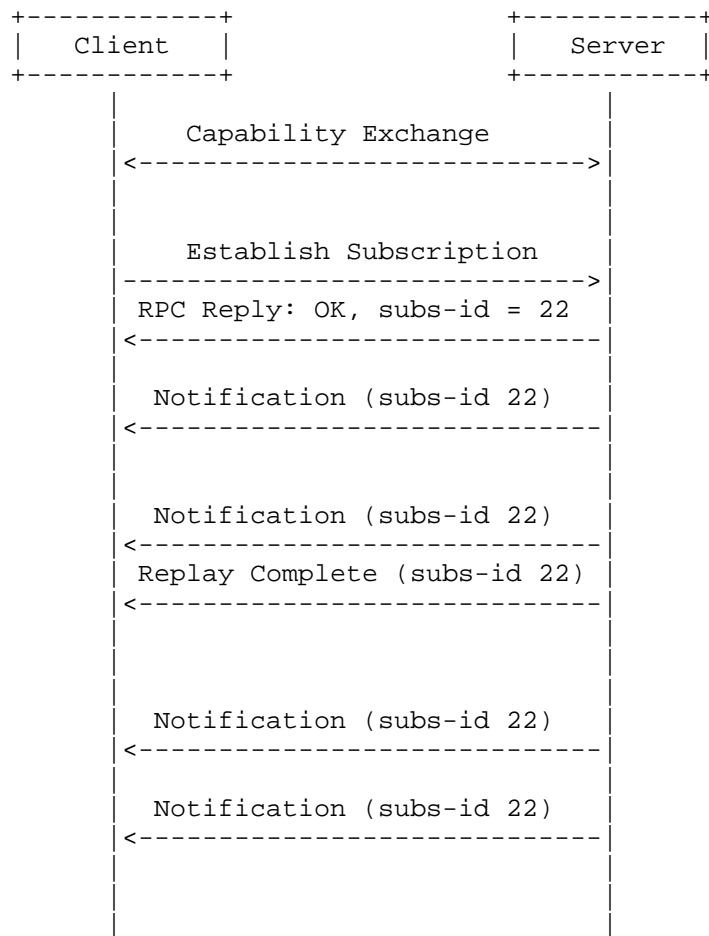


Figure 41: replayComplete notification

## 2.10.2. notificationComplete

```

<notification
  xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2007-09-01T10:00:00Z</eventTime>
  <notificationComplete
    xmlns="urn:ietf:params:xml:ns:netmod:notification"/>
</notification>

```

Figure 42: notificationComplete control plane notification

#### 2.10.2.1. Message Flow Example

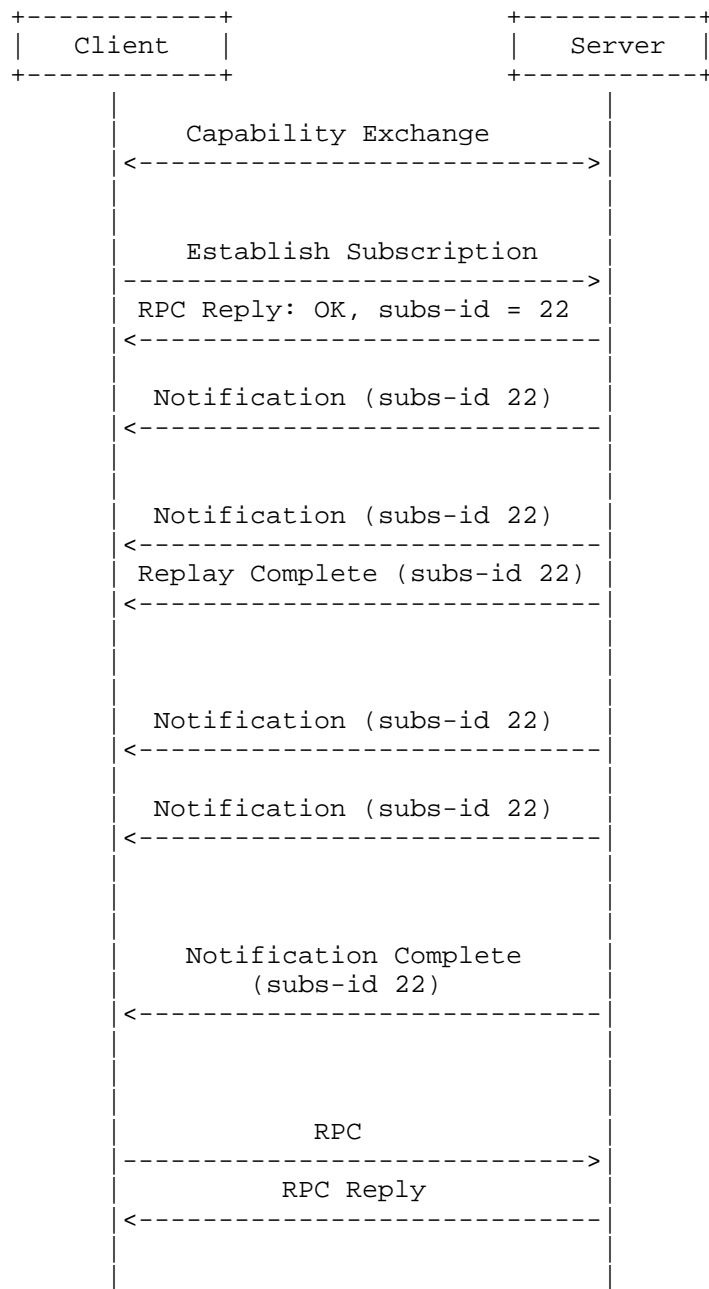


Figure 43: notificationComplete notification

## 2.10.3. subscription-started

```
<notification
  xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2007-09-01T10:00:00Z</eventTime>
  <subscription-started
    xmlns="urn:ietf:params:xml:ns:yang:ietf-event-notifications"/>
    <subscription-id>52</subscription-id>
    <filter netconf:type="xpath"
      xmlns:ex="http://example.com/event/1.0"
      select="/ex:event[ex:eventClass='fault' and
        (ex:severity='minor' or ex:severity='major'
          or ex:severity='critical')]">
    </subscription-started>
  </notification>
```

Figure 44: subscription-started control plane notification

The equivalent using JSON encoding would be:

```
<notification
  xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2007-09-01T10:00:00Z</eventTime>
  <notification-contents-json>
    {
      "notif-bis:subscription-started": {
        "subscription-id" : 52
        ((Open Item: express filter in json))
      }
    }
  </notification-contents-json>
</notification>
```

Figure 45: subscription-started control plane notification (JSON encoding)

## 2.10.4. subscription-modified

```
<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-event-notifications"/>
    <subscription-id>52</subscription-id>
    <filter netconf:type="xpath"
      xmlns:ex="http://example.com/event/1.0"
      select="/ex:event[ex:eventClass='fault']"/>
  </subscription-modified/>
</notification>
```

Figure 46: subscription-modified control plane notification

#### 2.10.5. subscription-terminated

```
<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-event-notifications"/>
    <subscription-id>52</subscription-id>
    <reason>subscription-deleted</reason>
  </subscription-terminated/>
</notification>
```

Figure 47: subscription-terminated control plane notification

#### 2.10.6. subscription-suspended

```
<notification
  xmlns=" urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2007-09-01T10:00:00Z</eventTime>
  <subscription-suspended
    xmlns="urn:ietf:params:xml:ns:yang:ietf-event-notifications"/>
    <subscription-id>52</subscription-id>
    <reason>internal-error</reason>
  </subscription-suspended/>
</notification>
```

Figure 48: subscription-suspended control plane notification

#### 2.10.7. subscription-resumed

```
<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-event-notifications"/>
    <subscription-id>52</subscription-id>
    <reason>internal-error</reason>
  </subscription-resumed/>
</notification>
```

Figure 49: subscription-resumed control plane notification

#### 2.10.7.1. Message Flow Examples

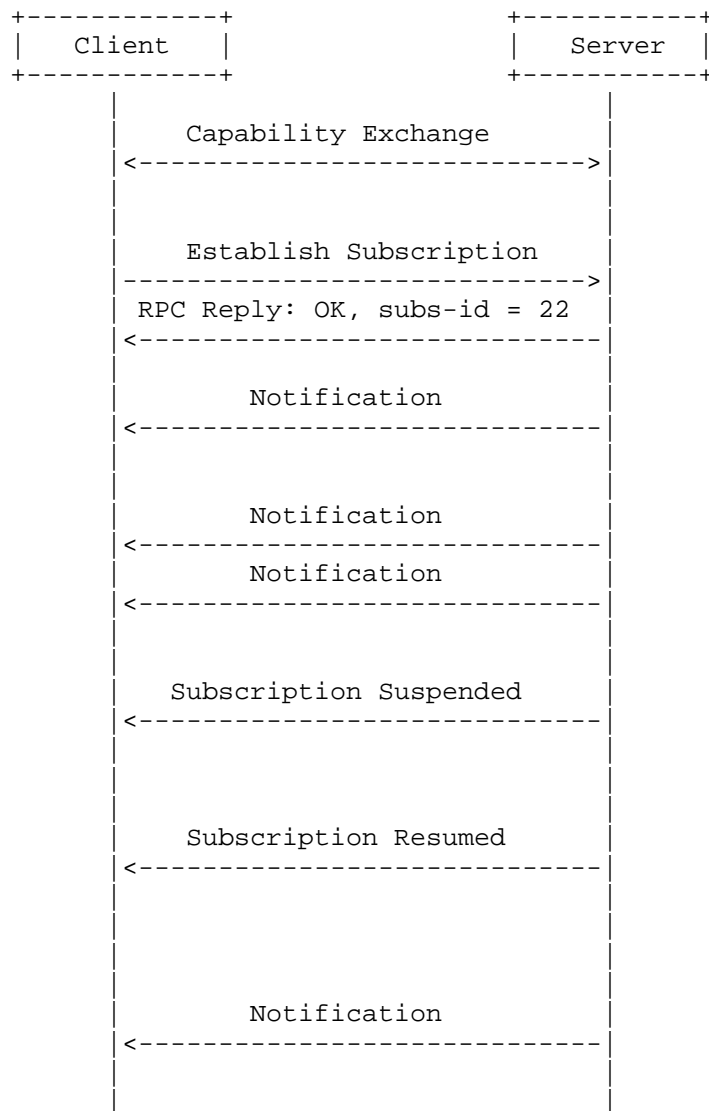
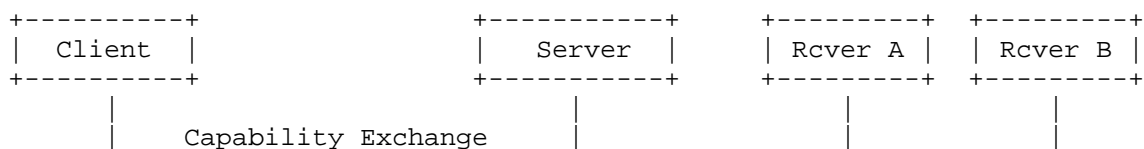


Figure 50: subscription-suspended and Resumed Notifications





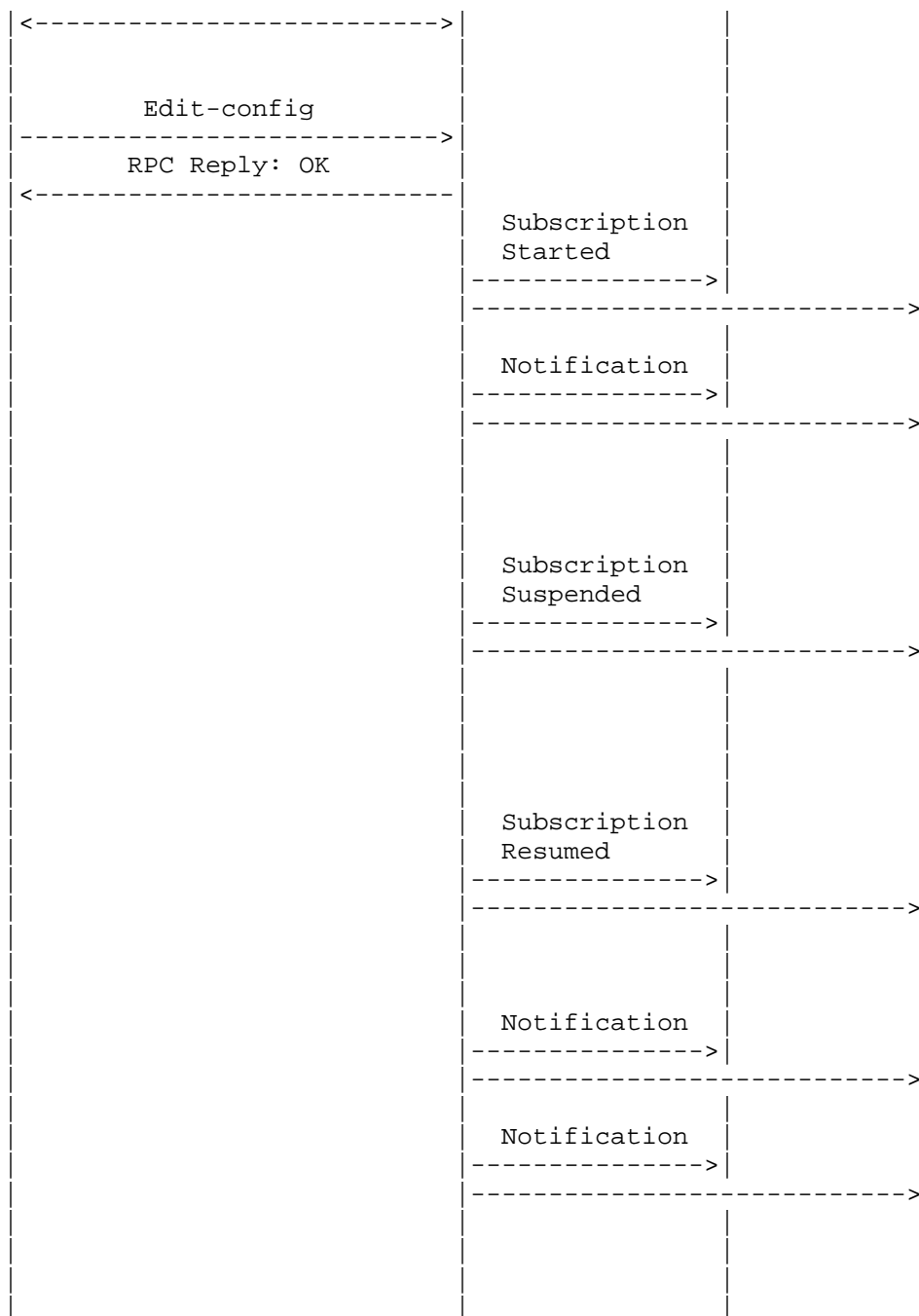


Figure 51: subscription-suspended and subscription-resumed notifications (configured subscriptions)

### 3. Backwards Compatibility

#### 3.1. Capabilities

Capabilities are advertised in messages sent by each peer during session establishment [RFC6241]. Servers supporting the features in this document must advertise both capabilities "urn:ietf:params:netconf:capability:notification:1.0" and "urn:ietf:params:netconf:capability:notification:1.1".

An example of a hello message by a server during session establishment would be:

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>
      urn:ietf:params:xml:ns:netconf:base:1.0
    </capability>
    <capability>
      urn:ietf:params:netconf:capability:startup:1.0
    </capability>
    <capability>
      urn:ietf:params:netconf:capability:notification:1.0
    </capability>
    <capability>
      urn:ietf:params:netconf:capability:notification:1.1
    </capability>
  </capabilities>
  <session-id>4</session-id>
</hello>
```

Figure 52: Hello message

Clients that only support [RFC5277] recognize capability "urn:ietf:params:netconf:capability:notification:1.0" and ignore capability "urn:ietf:params:netconf:capability:notification:1.1". This allows them interacting with the server as per [RFC5277]. Clients that support the features in this document recognize both capabilities. This allows them interacting with the server as per this document.

### 3.2. Stream Discovery

In order to maintain backwards compatibility, clients that only support [RFC5277] can retrieve the list of available event streams executing a <get> operation against the container `"/netconf/streams"`.

## 4. Security Considerations

The security considerations from the base NETCONF document [RFC6241] also apply to the notification capability.

The <notification> elements are never sent before the transport layer and the NETCONF layer, including capabilities exchange, have been established and the manager has been identified and authenticated.

A secure transport must be used and the server must ensure that the user has sufficient authorization to perform the function they are requesting against the specific subset of NETCONF content involved. When a <get> is received that refers to the content defined in this memo, clients should only be able to view the content for which they have sufficient privileges. <create-subscription> and <establish-subscription> operations can be considered like deferred <get>, and the content that different users can access may vary. This different access is reflected in the <notification> that different users are able to subscribe to.

The contents of notifications, as well as the names of event streams, may contain sensitive information and care should be taken to ensure that they are viewed only by authorized users. The NETCONF server MUST NOT include any content in a notification that the user is not authorized to view.

If a malicious or buggy NETCONF client sends a number of <create-subscription> requests, then these subscriptions accumulate and may use up system resources. In such a situation, subscriptions can be terminated by terminating the suspect underlying NETCONF sessions using the <kill-session> operation. If the client uses <establish-subscription>, the server can also suspend or terminate subscriptions with per-subscription granularity.

A subscription could be configured on another receiver's behalf, with the goal of flooding that receiver with updates. One or more publishers could be used to overwhelm a receiver, which doesn't even support subscriptions. Clients that do not want pushed data need only terminate or refuse any transport sessions from the publisher. In addition, the NETCONF Authorization Control Model [RFC6536] SHOULD be used to control and restrict authorization of subscription configuration. This control models permits specifying per-user

permissions to receive specific event notification types. The permissions are specified as a set of access control rules.

Note that streams can define additional authorization requirements. For instance, in [yang-push] each of the elements in its data plane notifications must also go through access control.

## 5. Acknowledgments

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

## 6. References

### 6.1. 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, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <<http://www.rfc-editor.org/info/rfc3339>>.
- [RFC5277] Chisholm, S. and H. Trevino, "NETCONF Event Notifications", RFC 5277, DOI 10.17487/RFC5277, July 2008, <<http://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, <<http://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, <<http://www.rfc-editor.org/info/rfc6241>>.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <<http://www.rfc-editor.org/info/rfc6536>>.

## 6.2. Informative References

### [call-home]

Watsen, K., "NETCONF Call Home and RESTCONF Call Home", December 2015, <<https://datatracker.ietf.org/doc/draft-ietf-netconf-call-home/>>.

### [event-notifications]

Clemm, A., Gonzalez Prieto, A., Voit, Eric., Nilsen-Nygaard, E., Tripathy, A., Chisholm, S., and H. Trevino, "Subscribing to Event Notifications", June 2016, <<https://datatracker.ietf.org/doc/draft-ietf-netconf-rfc5277bis/>>.

### [yang-push]

Clemm, A., Gonzalez Prieto, A., Voit, Eric., Tripathy, A., and E. Nilsen-Nygaard, "Subscribing to YANG datastore push updates", February 2016, <<https://datatracker.ietf.org/doc/draft-ietf-netconf-yang-push/>>.

## Appendix A. Issues that are currently being worked

(To be removed by RFC editor prior to publication)

- o NT1 - Express filter in JSON should be documented.

## Appendix B. Changes between revisions

(To be removed by RFC editor prior to publication)

### B.1. v00 to v01

- o D1 - Added Call Home in solution for configured subscriptions.
- o D2 - Clarified support for multiple subscription on a single session. No need to support multiple create-subscription.
- o D3 - Added mapping between terminology in [yang-push] and [RFC6241] (the one followed in this document).
- o D4 - Editorial improvements.

## Authors' Addresses

Alberto Gonzalez Prieto  
Cisco Systems

Email: albertgo@cisco.com

Alexander Clemm  
Sympotech

Email: alex@sympotech.com

Eric Voit  
Cisco Systems

Email: evoit@cisco.com

Einar Nilsen-Nygaard  
Cisco Systems

Email: einarnn@cisco.com

Ambika Prasad Tripathy  
Cisco Systems

Email: ambtripa@cisco.com

Sharon Chisholm  
Ciena

Email: schishol@ciena.com

Hector Trevino  
Cisco Systems

Email: htrevino@cisco.com

NETCONF  
Internet-Draft  
Intended status: Standards Track  
Expires: November 20, 2019

E. Voit  
Cisco Systems  
A. Clemm  
Huawei  
A. Gonzalez Prieto  
Microsoft  
E. Nilsen-Nygaard  
A. Tripathy  
Cisco Systems  
May 19, 2019

Dynamic subscription to YANG Events and Datastores over NETCONF  
draft-ietf-netconf-netconf-event-notifications-22

#### Abstract

This document provides a Network Configuration Protocol (NETCONF) binding to the dynamic subscription capability of both subscribed notifications and YANG-Push.

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

#### 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 November 20, 2019.

#### Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

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

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

## Table of Contents

1. Introduction . . . . .	3
2. Terminology . . . . .	3
3. Compatibility with RFC-5277's create-subscription . . . . .	3
4. Mandatory XML, event stream and datastore support . . . . .	4
5. NETCONF connectivity and the Dynamic Subscriptions . . . . .	4
6. Notification Messages . . . . .	4
7. Dynamic Subscriptions and RPC Error Responses . . . . .	5
8. Security Considerations . . . . .	7
9. IANA Considerations . . . . .	7
10. Acknowledgments . . . . .	7
11. References . . . . .	7
11.1. Normative References . . . . .	7
11.2. Informative References . . . . .	8
Appendix A. Examples . . . . .	8
A.1. Event Stream Discovery . . . . .	8
A.2. Dynamic Subscriptions . . . . .	9
A.3. Subscription State Notifications . . . . .	14
A.4. Filter Examples . . . . .	15
Appendix B. Changes between revisions . . . . .	17
B.1. v21 to v22 . . . . .	17
B.2. v20 to v21 . . . . .	17
B.3. v19 to v20 . . . . .	17
B.4. v17 to v19 . . . . .	17
B.5. v16 to v17 . . . . .	17
B.6. v15 to v16 . . . . .	18
B.7. v14 to v15 . . . . .	18



B.8. v13 to v14	18
B.9. v11 to v13	18
B.10. v10 to v11	18
B.11. v09 to v10	18
B.12. v08 to v09	18
B.13. v07 to v08	18
B.14. v06 to v07	19
B.15. v05 to v06	19
B.16. v03 to v04	19
B.17. v01 to v03	19
B.18. v00 to v01	19
Authors' Addresses	19

## 1. Introduction

This document specifies the binding of a stream of events which form part of a dynamic subscription to the NETCONF protocol [RFC6241]. Dynamic subscriptions are defined in [I-D.draft-ietf-netconf-subscribed-notifications]. In addition, as [I-D.draft-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.

This document assumes that the reader is familiar with the terminology and concepts defined in [I-D.draft-ietf-netconf-subscribed-notifications].

## 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 MUST NOT 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] "rpc-error" element containing the "error-tag" value of "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] "rpc-error" element containing the "error-tag" value of "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

Management of dynamic subscriptions occurs via RPCs as defined in [I-D.draft-ietf-netconf-yang-push] and [I-D.draft-ietf-netconf-subscribed-notifications]. 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 "resync-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" is 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

When an RPC error occurs as defined in [I-D.draft-ietf-netconf-subscribed-notifications] Section 2.4.6 and [I-D.draft-ietf-netconf-yang-push] Appendix A, 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 with the value being a string that corresponds to an identity associated with the error. For the mechanisms specified in this document, this "error-tag" will come from one of two places. Either it will correspond to the error identities within [I-D.draft-ietf-netconf-subscribed-notifications] section 2.4.6 for general subscription errors:

error identity	uses error-tag
dscp-unavailable	invalid-value
encoding-unsupported	invalid-value
filter-unsupported	invalid-value
insufficient-resources	resource-denied
no-such-subscription	invalid-value
replay-unsupported	operation-not-supported

Or this "error-tag" will correspond to the error identities within [I-D.draft-ietf-netconf-yang-push] Appendix A.1 for subscription errors specific to YANG datastores:

error identity	uses error-tag
cant-exclude	operation-not-supported
datastore-not-subscribable	invalid-value
no-such-subscription-resync	invalid-value
on-change-unsupported	operation-not-supported
on-change-sync-unsupported	operation-not-supported
period-unsupported	invalid-value
update-too-big	too-big
sync-too-big	too-big
unchanging-selection	operation-failed

- o an "error-severity" of "error" (this MAY 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.draft-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	have base identity
-----	-----
establish-subscription	establish-subscription-error
modify-subscription	modify-subscription-error
delete-subscription	delete-subscription-error
kill-subscription	delete-subscription-error
resync-subscription	resync-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.draft-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 "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 "resync-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

This document does not introduce additional Security Considerations for dynamic subscriptions beyond those discussed in [I-D.draft-ietf-netconf-subscribed-notifications]. But there is one consideration worthy of more refinement based on the connection oriented nature of the NETCONF protocol. Specifically, if a buggy or compromised 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 in order to reclaim resources and preserve normal operation for the other subscriptions.

## 9. IANA Considerations

This document has no actions for IANA.

## 10. 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, Qin Wu, and Guangying Zheng.

## 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/](https://datatracker.ietf.org/doc/draft-ietf-netconf-subscribed-notifications/)>.
- [I-D.draft-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/](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. Additionally the subscription "id" values of 22, 23, and 39 used below are just examples. In production, the actual values of "id" may not be small integers.

### 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.

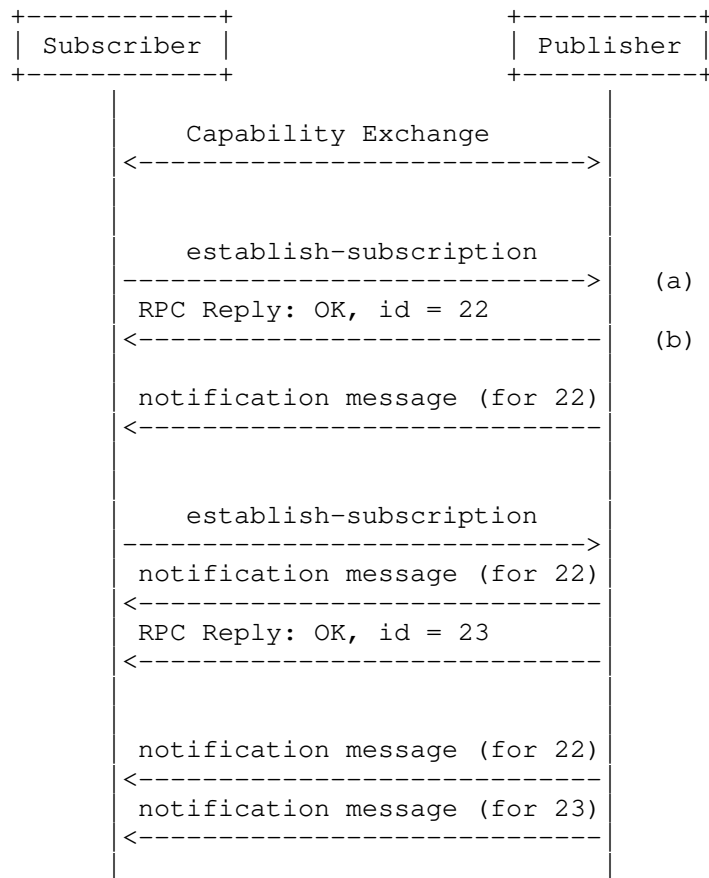


Figure 2: Multiple subscriptions over a NETCONF session

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-xpath-filter xmlns:ex="http://example.com/events">
      /ex:foo/
    </stream-xpath-filter>
    <stream>NETCONF</stream>
    <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>invalid-value</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.

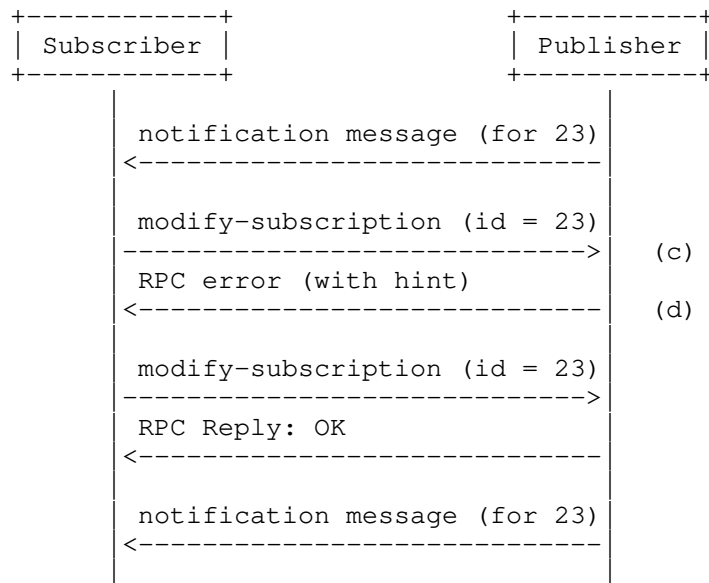


Figure 6: Interaction model for successful subscription modification

If the subscription being modified in Figure 6 is a datastore subscription as per [I-D.draft-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:ex="http://example.com/datastore">
      /ex:foo/ex: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>invalid-value</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>invalid-value</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. v21 to v22

- o Added "is".

### B.2. v20 to v21

- o Including Tom Petch's text to resolve the meaning of 'binding'.
- o A few small wording tweaks.

### B.3. v19 to v20

- o Notes to RFC editor removed, consideration moved under Figure 10 in SN.

### B.4. v17 to v19

- o Per Benjamin Kaduk's discuss on SN, adjusted IPR to pre5378Trust200902

### B.5. v16 to v17

- o During the SN YANG Doctor review, a suggestion was made to update the error-tags to make the mechanism work with embedded NETCONF and RESTCONF error reporting.
- o Minor text tweaks from review.

## B.6. v15 to v16

- o During the shepherd review, two clarifications were requested which do not impact the technical details of this document. These clarifications were: (a) further describing that dynamic subscriptions can have state change notifications, and (b) more details about the recommended text refinement desired for RFC6241.

## B.7. v14 to v15

- o Per Kent's request, added name attribute to artwork. This would be needed for an automated extraction.

## B.8. v13 to v14

- o Title change.

## B.9. 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.10. v10 to v11

- o Configured removed.

## B.11. v09 to v10

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

## B.12. 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.13. v07 to v08

- o Tweaks and clarification on :interleave.



## B.14. v06 to v07

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

## B.15. 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.16. 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.17. v01 to v03

- o Text simplifications throughout
- o v02 had no meaningful changes

## B.18. 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](mailto: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: September 14, 2017

K. Watsen  
Juniper Networks  
J. Schoenwaelder  
Jacobs University Bremen  
March 13, 2017

RESTCONF Client and Server Models  
draft-ietf-netconf-restconf-client-server-02

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 "2017-03-13" --> the publication date of this draft

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

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

#### 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 <http://datatracker.ietf.org/drafts/current/>.

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

This Internet-Draft will expire on September 14, 2017.

#### Copyright Notice

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

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

#### Table of Contents

1. Introduction . . . . .	3
1.1. Terminology . . . . .	3
1.2. Tree Diagrams . . . . .	3
2. The RESTCONF Client Model . . . . .	4
2.1. Tree Diagram . . . . .	4
2.2. Example Usage . . . . .	6
2.3. YANG Model . . . . .	8
3. The RESTCONF Server Model . . . . .	16
3.1. Tree Diagram . . . . .	16
3.2. Example Usage . . . . .	18

3.3. YANG Model . . . . .	20
4. Security Considerations . . . . .	29
5. IANA Considerations . . . . .	30
5.1. The IETF XML Registry . . . . .	30
5.2. The YANG Module Names Registry . . . . .	30
6. Acknowledgements . . . . .	30
7. References . . . . .	31
7.1. Normative References . . . . .	31
7.2. Informative References . . . . .	31
Appendix A. Change Log . . . . .	33
A.1. server-model-09 to 00 . . . . .	33
A.2. 00 to 01 . . . . .	33
A.3. 01 to 02 . . . . .	33
Appendix B. Open Issues . . . . .	33
Authors' Addresses . . . . .	33

## 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 [RFC5246] transport protocol with both standard RESTCONF and RESTCONF Call Home connections [RFC8071].

### 1.1. Terminology

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

### 1.2. Tree Diagrams

A simplified graphical representation of the data models is used in this document. The meaning of the symbols in these diagrams is as follows:

- o Brackets "[" and "]" enclose list keys.
- o Braces "{" and "}" enclose feature names, and indicate that the named feature must be present for the subtree to be present.
- o Abbreviations before data node names: "rw" means configuration (read-write) and "ro" state data (read-only).
- o Symbols after data node names: "?" means an optional node, "!" means a presence container, and "\*" denotes a list and leaf-list.

- o Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- o Ellipsis ("...") stands for contents of subtrees that are not shown.

## 2. The RESTCONF Client Model

EDITOR NOTE: Please ignore this section, it is incomplete.

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

This model supports the TLS transport protocol using the TLS client groupings defined in [I-D.ietf-netconf-tls-client-server].

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

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

```

module: ietf-restconf-client
  +--rw restconf-client
    +--rw initiate {initiate}?
      +--rw restconf-server* [name]
        +--rw name string
        +--rw (transport)
          +--:(tls) {tls-initiate}?
            +--rw tls
              +--rw endpoints
                +--rw endpoint* [name]
                  +--rw name string
                  +--rw address inet:host
                  +--rw port? inet:port-number
              +--rw server-auth
                +--rw trusted-ca-certs? leafref
                +--rw trusted-server-certs? leafref
              +--rw client-auth
                +--rw (auth-type)?
                  +--:(certificate)
                    +--rw certificate? leafref
              +--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 idle-timeout?   uint32
    |   +--rw keep-alives
    |       +--rw max-wait?       uint16
    |       +--rw max-attempts?   uint8
    +--:(periodic-connection)
    |   +--rw periodic!
    |   +--rw idle-timeout?       uint16
    |   +--rw reconnect-timeout?   uint16
+--rw reconnect-strategy
    +--rw start-with?   enumeration
    +--rw max-attempts?   uint8
+--rw listen {listen}?
    +--rw max-sessions?   uint16
    +--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 server-auth
            |   +--rw trusted-ca-certs?   leafref
            |   +--rw trusted-server-certs? leafref
            +--rw client-auth
            |   +--rw (auth-type)?
            |   |   +--:(certificate)
            |   |   +--rw certificate?   leafref
            +--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 2.2 of [I-D.ietf-netconf-keystore].



```
<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>
      <tls>
        <endpoints>
          <endpoint>
            <name>corp-fw1.example.com</name>
            <address>corp-fw1.example.com</address>
          </endpoint>
          <endpoint>
            <name>corp-fw2.example.com</name>
            <address>corp-fw2.example.com</address>
          </endpoint>
        </endpoints>
        <server-auth>
          <trusted-server-certs>deployment-specific-ca-certs</trusted-server-cer
ts>
        </server-auth>
        <client-auth>
          <certificate>tls-ec-cert</certificate>
        </client-auth>
      </tls>
    </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>
        <server-auth>
          <trusted-ca-certs>deployment-specific-ca-certs</trusted-ca-certs>
          <trusted-server-certs>explicitly-trusted-server-certs</trusted-server-
certs>
        </server-auth>
        <client-auth>
          <certificate>tls-ec-cert</certificate>
        </client-auth>
      </tls>
    </endpoint>
  </listen>
</restconf-client>
```

### 2.3. YANG Model

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

```
<CODE BEGINS> file "ietf-restconf-client@2017-03-13.yang"

module ietf-restconf-client {
  yang-version 1.1;

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

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

  import ietf-tls-client {
    prefix ts;
    revision-date 2017-03-13; // stable grouping definitions
    reference
      "RFC ZZZZ: TLS Client and Server Models";
  }

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

  contact
    "WG Web:  <http://tools.ietf.org/wg/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) 2014 IETF Trust and the persons identified as
    authors of the code. All rights reserved.

    Redistribution and use in source and binary forms, with or
```

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

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

```
revision "2017-03-13" {
  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 {
  description
    "The 'tls-initiate' feature indicates that the RESTCONF client
    supports initiating TLS connections to RESTCONF servers.";
  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 {
  description
    "The 'tls-listen' feature indicates that the RESTCONF client
    supports opening a port to listen for incoming RESTCONF
    server call-home TLS connections.";
  reference
    "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
}
```

```
}

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

  container initiate {
    if-feature initiate;
    description
      "Configures client initiating underlying TCP connections.";
    list restconf-server {
      key name;
      description
        "List of RESTCONF servers the RESTCONF client is to initiate
        connections to.";
      leaf name {
        type string;
        description
          "An arbitrary name for the RESTCONF server.";
      }
      choice transport {
        mandatory true;
        description
          "Selects between available transports.";

        case tls {
          if-feature tls-initiate;
          container tls {
            description
              "Specifies TLS-specific transport configuration.";
            uses endpoints-container {
              refine endpoints/endpoint/port {
                default 443;
              }
            }
            uses ts:tls-client-grouping;
          }
        } // end tls
      } // end transport
    }

    container connection-type {
      description
        "Indicates the kind of connection to use.";
      choice connection-type {
        description
          "Selects between available connection types.";
        case persistent-connection {
```

```
container persistent {
  presence true;
  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.";
  leaf idle-timeout {
    type uint32;
    units "seconds";
    default 86400; // one day;
    description
      "Specifies the maximum number of seconds that a
      a RESTCONF session may remain idle. A RESTCONF
      session will be dropped if it is idle for an
      interval longer than this number of seconds.
      If set to zero, then the client will never drop
      a session because it is idle. Sessions that
      have a notification subscription active are
      never dropped.";
  }
}
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.";
  reference
    "RFC 8071: NETCONF Call Home and RESTCONF Call
    Home, Section 3.1, item S6";
  leaf max-wait {
    type uint16 {
      range "1..max";
    }
    units seconds;
    default 30;
    description
      "Sets the amount of time in seconds after which
      if no data has been received from the 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 true;
        description
            "Periodically connect to the RESTCONF server, so that
            the RESTCONF server may deliver messages pending for
            the RESTCONF client. The RESTCONF server must close
            the connection when it is ready to release it. Once
            the connection has been closed, the RESTCONF client
            will restart its timer until the next connection.";
        leaf idle-timeout {
            type uint16;
            units "seconds";
            default 300; // five minutes
            description
                "Specifies the maximum number of seconds that a
                a RESTCONF session may remain idle. A RESTCONF
                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.";
        }
        leaf reconnect-timeout {
            type uint16 {
                range "1..max";
            }
            units minutes;
            default 60;
            description
                "Sets the maximum amount of unconnected time the
                RESTCONF client will wait before re-establishing
                a connection to the RESTCONF server. The RESTCONF
                client may initiate a connection before this
                time if desired (e.g., to set configuration).";
        }
    }
}
```

```

    }
  }
}
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.";
      }
    }
  }
  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 restconf-server
} // end initiate

```

```
container listen {
  if-feature listen;
  description
    "Configures client accepting call-home TCP connections.";

  leaf max-sessions {
    type uint16;
    default 0;
    description
      "Specifies the maximum number of concurrent sessions
       that can be active at one time. The value 0 indicates
       that no artificial session limit should be used.";
  }

  leaf idle-timeout {
    type uint16;
    units "seconds";
    default 3600; // one hour
    description
      "Specifies the maximum number of seconds that a RESTCONF
       session may remain idle. A RESTCONF 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;
  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.";
    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 for call-home connections.";
    }
    leaf port {
        type inet:port-number;
        default 4336;
        description
            "The port number to listen for call-home connections.";
    }
    uses ts:tls-client-grouping;
}
} // end transport
} // end endpoint
} // end listen
} // end restconf-client
```

```
grouping endpoints-container {
    description
        "This grouping is used to configure a set of RESTCONF servers
        a RESTCONF client may initiate connections to.";
    container endpoints {
        description
            "Container for the list of endpoints.";
        list endpoint {
            key name;
            unique "address port";
            min-elements 1;
            ordered-by user;
            description
                "A non-empty user-ordered list of endpoints for this RESTCONF
                client to try to connect to. Defining more than one enables
                high-availability.";
            leaf name {
                type string;
                description
                    "An arbitrary name for this endpoint.";
            }
            leaf address {
                type inet:host;
                mandatory true;
                description
                    "The IP address or hostname of the endpoint. If a
                    hostname is configured and the DNS resolution results
                    in more than one IP address, the RESTCONF client
                    will process the IP addresses as if they had been
```

```

        explicitly configured in place of the hostname.";
    }
    leaf port {
        type inet:port-number;
        description
            "The IP port for this endpoint. The RESTCONF client will
             use the IANA-assigned well-known port (set via a refine
             statement when uses) if no value is specified.";
    }
}
}
```

&lt;CODE ENDS&gt;

### 3. The RESTCONF Server Model

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

This model supports the TLS transport protocol using the TLS server groupings defined in [I-D.ietf-netconf-tls-client-server].

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

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

```

module: ietf-restconf-server
  +--rw restconf-server
    +--rw listen {listen}?
      |
      | +--rw max-sessions?      uint16
      | +--rw endpoint* [name]
      | |   +--rw name          string
      | |   +--rw (transport)
      | |   |   +--:(tls) {tls-listen}?
      | |   |   |   +--rw tls
      | |   |   |   |   +--rw address?      inet:ip-address
      | |   |   |   |   +--rw port?         inet:port-number
      | |   |   |   |   +--rw certificates
      | |   |   |   |   |   +--rw certificate* [name]
      | |   |   |   |
      | |   |   |   +--rw (other)
      | |   |   |
      | |   |   +--rw (other)
      | |   +--rw (other)
      | +--rw (other)
      +--rw (other)

```

```

|         |--rw name      leafref
+--rw client-auth
|   |--rw trusted-ca-certs?      leafref
|   |--rw trusted-client-certs?  leafref
|   |--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 (transport)
|   +--:(tls) {tls-call-home}?
|   |   |--rw tls
|   |   |   |--rw endpoints
|   |   |   |   |--rw endpoint* [name]
|   |   |   |   |   |--rw name      string
|   |   |   |   |   |--rw address   inet:host
|   |   |   |   |   |--rw port?     inet:port-number
|   |   |   |--rw certificates
|   |   |   |   |--rw certificate* [name]
|   |   |   |   |   |--rw name      leafref
|   |   |--rw client-auth
|   |   |   |--rw trusted-ca-certs?      leafref
|   |   |   |--rw trusted-client-certs?  leafref
|   |   |   |--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 reconnect-timeout?    uint16
+--rw reconnect-strategy
  |--rw start-with?      enumeration
  |--rw max-attempts?    uint8

```

### 3.2. Example Usage

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

This example is consistent with the examples presented in Section 2.2 of [I-D.ietf-netconf-keystore].

```

<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">

  <!-- listening for TLS (HTTPS) connections -->
  <listen>
    <endpoint>
      <name>netconf/tls</name>
      <tls>
        <address>11.22.33.44</address>
        <certificates>
          <certificate>
            <name>tls-ec-cert</name>
          </certificate>
        </certificates>
        <client-auth>
          <trusted-ca-certs>deployment-specific-ca-certs</trusted-ca-certs>
          <trusted-client-certs>explicitly-trusted-client-certs</trusted-client-
certs>
        <cert-maps>
          <cert-to-name>
            <id>1</id>
            <fingerprint>11:0A:05:11:00</fingerprint>
            <map-type>x509c2n:san-any</map-type>
          </cert-to-name>
          <cert-to-name>
            <id>2</id>
            <fingerprint>B3:4F:A1:8C:54</fingerprint>
            <map-type>x509c2n:specified</map-type>

```

```

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

<!-- calling home to a RESTCONF client -->
<call-home>
  <restconf-client>
    <name>config-manager</name>
    <tls>
      <endpoints>
        <endpoint>
          <name>east-data-center</name>
          <address>22.33.44.55</address>
        </endpoint>
        <endpoint>
          <name>west-data-center</name>
          <address>33.44.55.66</address>
        </endpoint>
      </endpoints>
      <certificates>
        <certificate>
          <name>tls-ec-cert</name>
        </certificate>
      </certificates>
      <client-auth>
        <trusted-ca-certs>deployment-specific-ca-certs</trusted-ca-certs>
        <trusted-client-certs>explicitly-trusted-client-certs</trusted-client-
certs>
      <cert-maps>
        <cert-to-name>
          <id>1</id>
          <fingerprint>11:0A:05:11:00</fingerprint>
          <map-type>x509c2n:san-any</map-type>
        </cert-to-name>
        <cert-to-name>
          <id>2</id>
          <fingerprint>B3:4F:A1:8C:54</fingerprint>
          <map-type>x509c2n:specified</map-type>
          <name>scooby-doo</name>
        </cert-to-name>
      </cert-maps>
    </client-auth>
  </tls>
  <connection-type>
    <periodic>

```

```
        <idle-timeout>300</idle-timeout>
        <reconnect-timeout>60</reconnect-timeout>
    </periodic>
</connection-type>
<reconnect-strategy>
    <start-with>last-connected</start-with>
    <max-attempts>3</max-attempts>
</reconnect-strategy>
</restconf-client>
</call-home>

</restconf-server>
```

### 3.3. YANG Model

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

```
<CODE BEGINS> file "ietf-restconf-server@2017-03-13.yang"

module ietf-restconf-server {
  yang-version 1.1;

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

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

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

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

  import ietf-tls-server {
    prefix ts;
    revision-date 2017-03-13; // stable grouping definitions
```

```
reference
  "RFC ZZZZ: TLS Client and Server Models";
}

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

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

  WG Chair: Mehmet Ersue
              <mailto:mehmet.ersue@nsn.com>

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

  Editor: Kent Watsen
            <mailto:kwatsen@juniper.net>";

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

  Copyright (c) 2014 IETF Trust and the persons identified as
  authors of the code. All rights reserved.

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

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

revision "2017-03-13" {
  description
    "Initial version";
  reference
    "RFC XXXX: RESTCONF Client and Server Models";
}

// Features
```

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

feature tls-listen {
  description
    "The 'tls-listen' feature indicates that the RESTCONF server
    supports opening a port to listen for incoming RESTCONF
    client connections.";
  reference
    "RFC XXXX: RESTCONF Protocol";
}

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

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

feature client-cert-auth {
  description
    "The client-cert-auth feature indicates that the RESTCONF
    server supports the ClientCertificate authentication scheme.";
  reference
    "RFC ZZZZ: Client Authentication over New TLS Connection";
}

// top-level container
container restconf-server {
  description
    "Top-level container for RESTCONF server configuration.";

  container listen {
```



```
if-feature listen;
description
  "Configures listen behavior";
leaf max-sessions {
  type uint16;
  default 0;    // should this be 'max'?
  description
    "Specifies the maximum number of concurrent sessions
     that can be active at one time. The value 0 indicates
     that no artificial session limit should be used.";
}
list endpoint {
  key name;
  description
    "List of endpoints to listen for RESTCONF connections.";
  leaf name {
    type string;
    description
      "An arbitrary name for the RESTCONF listen endpoint.";
  }
  choice transport {
    mandatory true;
    description
      "Selects between available transports.";
    case tls {
      if-feature tls-listen;
      container tls {
        description
          "TLS-specific listening configuration for inbound
           connections.";
        leaf address {
          type inet:ip-address;
          description
            "The IP address of the interface to listen on. The
             TLS server will listen on all interfaces if no value
             is specified. Please note that some addresses have
             special meanings (e.g., '0.0.0.0' and ':::').";
        }
        leaf port {
          type inet:port-number;
          default 443;
          description
            "The local port number on this interface the TLS server
             listens on.";
        }
      }
      uses ts:tls-server-grouping {
        augment "client-auth" {
          description
```

```

        "Augments in the cert-to-name structure.";
        uses cert-maps-grouping;
    }
}
}
}
}
}

container call-home {
    if-feature call-home;
    description
        "Configures call-home behavior";
    list restconf-client {
        key name;
        description
            "List of RESTCONF clients the RESTCONF server is to
            initiate call-home connections to.";
        leaf name {
            type string;
            description
                "An arbitrary name for the remote RESTCONF client.";
        }
        choice transport {
            mandatory true;
            description
                "Selects between TLS and any transports augmented in.";
            case tls {
                if-feature tls-call-home;
                container tls {
                    description
                        "Specifies TLS-specific call-home transport
                        configuration.";
                    uses endpoints-container {
                        refine endpoints/endpoint/port {
                            default 4336;
                        }
                    }
                    uses ts:tls-server-grouping {
                        augment "client-auth" {
                            description
                                "Augments in the cert-to-name structure.";
                                uses cert-maps-grouping;
                            }
                    }
                }
            }
        }
    }
}

```

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

          This connection type minimizes any RESTCONF client
          to RESTCONF server data-transfer delay, albeit at
          the expense of holding resources longer.";
      }
    }
  }
  container keep-alives {
    description
      "Configures the keep-alive policy, to proactively
      test the aliveness of the TLS client. An
      unresponsive TLS client will be dropped after
      approximately (max-attempts * max-wait)
      seconds.";
    reference
      "RFC 8071: NETCONF Call Home and RESTCONF Call
      Home, Section 3.1, item S6";
    leaf max-wait {
      type uint16 {
        range "1..max";
      }
      units seconds;
      default 30;
      description
        "Sets the amount of time in seconds after which
        if no data has been received from the TLS
        client, a TLS-level message will be sent to
        test the aliveness of the TLS client.";
    }
    leaf max-attempts {
      type uint8;
      default 3;
      description
        "Sets the maximum number of sequential keep-alive
```

```

        messages that can fail to obtain a response from
        the TLS client before assuming the TLS client is
        no longer alive.";
    }
}
}
}
case periodic-connection {
  container periodic {
    presence true;
    description
      "Periodically connect to the RESTCONF client, so that
      the RESTCONF client may deliver messages pending for
      the RESTCONF server. The client must close the
      connection when it's ready to release it. Once the
      connection has been closed, the server will restart
      its timer until the next connection.";
    leaf reconnect-timeout {
      type uint16 {
        range "1..max";
      }
      units minutes;
      default 60;
      description
        "The maximum amount of unconnected time the
        RESTCONF server will wait before re-establishing
        a connection to the RESTCONF client. The
        RESTCONF server may initiate a connection to
        the RESTCONF client before this time if desired
        (e.g., to deliver a notification).";
    }
  }
}
}
}
}
container reconnect-strategy {
  description
    "The reconnection strategy directs how a RESTCONF server
    reconnects to a RESTCONF client after after discovering
    its connection to the client has dropped, even if due to
    a reboot. The RESTCONF server starts with the specified
    endpoint and tries to connect to it max-attempts times
    before trying the next endpoint in the list (round
    robin).";
  leaf start-with {
    type enumeration {
      enum first-listed {
        description

```

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

```
grouping cert-maps-grouping {
    description
        "A grouping that defines a container around the
        cert-to-name structure defined in RFC 7407.";
    container cert-maps {
        uses x509c2n:cert-to-name;
        description
            "The cert-maps container is used by a TLS-based RESTCONF
            server to map the RESTCONF client's presented X.509
            certificate to a RESTCONF username.  If no matching and
            valid cert-to-name list entry can be found, then the
            RESTCONF server MUST close the connection, and MUST NOT
            accept RESTCONF messages over it.";
    }
}
```

```
    reference
      "RFC XXXX: The RESTCONF Protocol";
  }
}

grouping endpoints-container {
  description
    "This grouping is used by tls container for call-home
    configurations.";
  container endpoints {
    description
      "Container for the list of endpoints.";
    list endpoint {
      key name;
      unique "address port";
      min-elements 1;
      ordered-by user;
      description
        "User-ordered list of endpoints for this RESTCONF client.
        Defining more than one enables high-availability.";
      leaf name {
        type string;
        description
          "An arbitrary name for this endpoint.";
      }
      leaf address {
        type inet:host;
        mandatory true;
        description
          "The IP address or hostname of the endpoint.  If a
          hostname is configured and the DNS resolution results
          in more than one IP address, the RESTCONF server
          will process the IP addresses as if they had been
          explicitly configured in place of the hostname.";
      }
      leaf port {
        type inet:port-number;
        description
          "The IP port for this endpoint. The RESTCONF server will
          use the IANA-assigned well-known port if no value is
          specified.";
      }
    }
  }
}
```

```
}
```

```
<CODE ENDS>
```

#### 4. Security Considerations

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) [RFC6536] 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:

NONE

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 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 [RFC7950]. Following the format in [RFC7950], the the following registrations are requested:

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

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

## 6. Acknowledgements

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

Juergen Schoenwaelder and was partly funded by Flamingo, a Network of Excellence project (ICT-318488) supported by the European Commission under its Seventh Framework Programme.



## 7. References

### 7.1. Normative References

- [I-D.ietf-netconf-keystore]  
Watsen, K. and G. Wu, "Keystore Model", draft-ietf-netconf-keystore-00 (work in progress), October 2016.
- [I-D.ietf-netconf-tls-client-server]  
Watsen, K., "TLS Client and Server Models", draft-ietf-netconf-tls-client-server-01 (work in progress), November 2016.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <<http://www.rfc-editor.org/info/rfc6536>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<http://www.rfc-editor.org/info/rfc6991>>.
- [RFC7407] Bjorklund, M. and J. Schoenwaelder, "A YANG Data Model for SNMP Configuration", RFC 7407, DOI 10.17487/RFC7407, December 2014, <<http://www.rfc-editor.org/info/rfc7407>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<http://www.rfc-editor.org/info/rfc7950>>.

### 7.2. Informative References

- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<http://www.rfc-editor.org/info/rfc3688>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<http://www.rfc-editor.org/info/rfc5246>>.

- [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, <<http://www.rfc-editor.org/info/rfc6241>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<http://www.rfc-editor.org/info/rfc8040>>.
- [RFC8071] Watsen, K., "NETCONF Call Home and RESTCONF Call Home", RFC 8071, DOI 10.17487/RFC8071, February 2017, <<http://www.rfc-editor.org/info/rfc8071>>.

## Appendix A. Change Log

## A.1. server-model-09 to 00

- o This draft was split out from draft-ietf-netconf-server-model-09.
- o Added in new features 'listen' and 'call-home' so future transports can be augmented in.

## A.2. 00 to 01

- o Renamed "keychain" to "keystore".

## A.3. 01 to 02

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

## Appendix B. Open Issues

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

## Authors' Addresses

Kent Watsen  
Juniper Networks

EMail: [kwatsen@juniper.net](mailto:kwatsen@juniper.net)

Juergen Schoenwaelder  
Jacobs University Bremen

EMail: [j.schoenwaelder@jacobs-university.de](mailto:j.schoenwaelder@jacobs-university.de)

NETCONF Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: 8 September 2022

K. Watsen  
Watsen Networks  
7 March 2022

RESTCONF Client and Server Models  
draft-ietf-netconf-restconf-client-server-25

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

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

- \* AAAA --> the assigned RFC value for draft-ietf-netconf-crypto-types
- \* BBBB --> the assigned RFC value for draft-ietf-netconf-trust-anchors
- \* CCCC --> the assigned RFC value for draft-ietf-netconf-keystore
- \* DDDD --> the assigned RFC value for draft-ietf-netconf-tcp-client-server
- \* EEEE --> the assigned RFC value for draft-ietf-netconf-ssh-client-server
- \* FFFF --> the assigned RFC value for draft-ietf-netconf-tls-client-server
- \* GGGG --> the assigned RFC value for draft-ietf-netconf-http-client-server

\* HHHH --> the assigned RFC value for draft-ietf-netconf-netconf-client-server

\* IIII --> the assigned RFC value for this draft

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

\* 2022-03-07 --> the publication date of this draft

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

\* Appendix A. Change Log

#### Status of This Memo

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

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

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

This Internet-Draft will expire on 8 September 2022.

#### Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

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

#### Table of Contents

1. Introduction . . . . .	4
---------------------------	---

1.1.	Relation to other RFCs . . . . .	4
1.2.	Specification Language . . . . .	6
1.3.	Adherence to the NMDA . . . . .	6
1.4.	Conventions . . . . .	6
2.	The "ietf-restconf-client" Module . . . . .	6
2.1.	Data Model Overview . . . . .	6
2.2.	Example Usage . . . . .	11
2.3.	YANG Module . . . . .	15
3.	The "ietf-restconf-server" Module . . . . .	25
3.1.	Data Model Overview . . . . .	25
3.2.	Example Usage . . . . .	30
3.3.	YANG Module . . . . .	34
4.	Security Considerations . . . . .	46
4.1.	The "ietf-restconf-client" YANG Module . . . . .	46
4.2.	The "ietf-restconf-server" YANG Module . . . . .	47
5.	IANA Considerations . . . . .	47
5.1.	The "IETF XML" Registry . . . . .	47
5.2.	The "YANG Module Names" Registry . . . . .	48
6.	References . . . . .	48
6.1.	Normative References . . . . .	48
6.2.	Informative References . . . . .	49
Appendix A.	Change Log . . . . .	51
A.1.	00 to 01 . . . . .	51
A.2.	01 to 02 . . . . .	51
A.3.	02 to 03 . . . . .	51
A.4.	03 to 04 . . . . .	51
A.5.	04 to 05 . . . . .	52
A.6.	05 to 06 . . . . .	52
A.7.	06 to 07 . . . . .	52
A.8.	07 to 08 . . . . .	52
A.9.	08 to 09 . . . . .	52
A.10.	09 to 10 . . . . .	53
A.11.	10 to 11 . . . . .	53
A.12.	11 to 12 . . . . .	53
A.13.	12 to 13 . . . . .	53
A.14.	13 to 14 . . . . .	54
A.15.	14 to 15 . . . . .	54
A.16.	15 to 16 . . . . .	54
A.17.	16 to 17 . . . . .	54
A.18.	17 to 18 . . . . .	55
A.19.	18 to 19 . . . . .	55
A.20.	19 to 20 . . . . .	55
A.21.	20 to 21 . . . . .	55
A.22.	21 to 22 . . . . .	55
A.23.	22 to 23 . . . . .	55
A.24.	23 to 24 . . . . .	55
A.25.	24 to 25 . . . . .	56
Acknowledgements	. . . . .	56

Author's Address . . . . . 56

## 1. Introduction

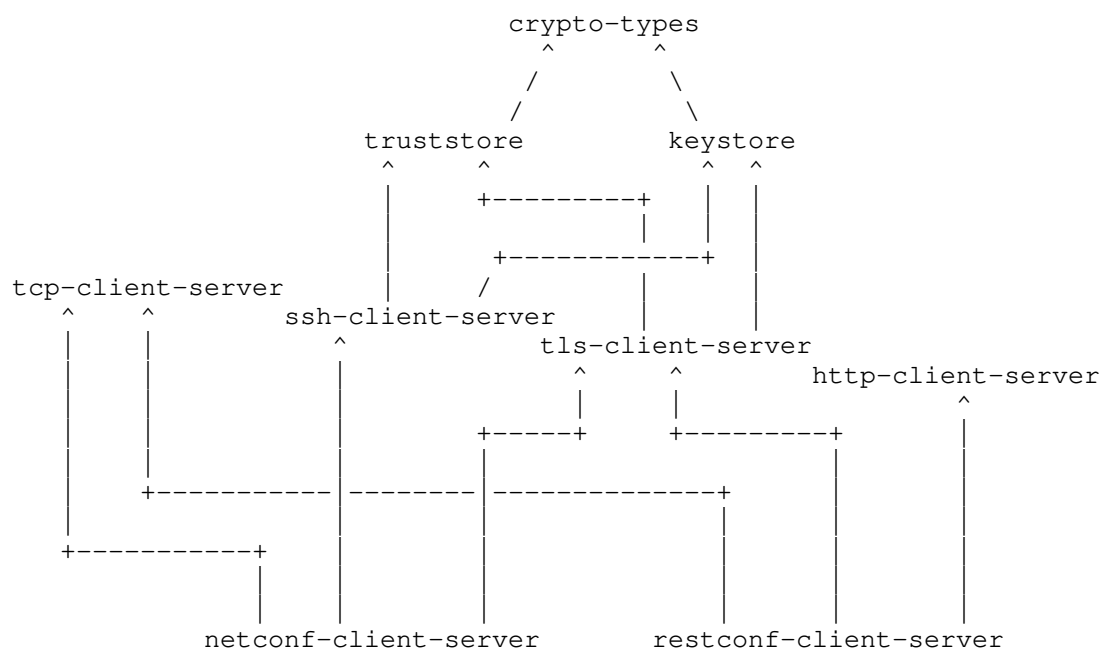
This document defines two YANG [RFC7950] modules, one module to configure a RESTCONF client and the other module to configure a RESTCONF server [RFC8040]. Both modules support the TLS [RFC8446] transport protocol with both standard RESTCONF and RESTCONF Call Home connections [RFC8071].

### 1.1. Relation to other RFCs

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

The modules have been defined in a modular fashion to enable their use by other efforts, some of which are known to be in progress at the time of this writing, with many more expected to be defined in time.

The normative dependency relationship between the various RFCs in the collection is presented in the below diagram. The labels in the diagram represent the primary purpose provided by each RFC. Hyperlinks to each RFC are provided below the diagram.



Label in Diagram	Originating RFC
crypto-types	[I-D.ietf-netconf-crypto-types]
truststore	[I-D.ietf-netconf-trust-anchors]
keystore	[I-D.ietf-netconf-keystore]
tcp-client-server	[I-D.ietf-netconf-tcp-client-server]
ssh-client-server	[I-D.ietf-netconf-ssh-client-server]
tls-client-server	[I-D.ietf-netconf-tls-client-server]
http-client-server	[I-D.ietf-netconf-http-client-server]
netconf-client-server	[I-D.ietf-netconf-netconf-client-server]
restconf-client-server	[I-D.ietf-netconf-restconf-client-server]

Table 1: Label to RFC Mapping



## 1.2. Specification Language

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

## 1.3. Adherence to the NMDA

This document is compliant with the Network Management Datastore Architecture (NMDA) [RFC8342]. For instance, as described in [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore], trust anchors and keys installed during manufacturing are expected to appear in <operational>.

## 1.4. Conventions

Various examples used in this document use a placeholder value for binary data that has been base64 encoded (e.g., "BASE64VALUE="). This placeholder value is used as real base64 encoded structures are often many lines long and hence distracting to the example being presented.

## 2. The "ietf-restconf-client" Module

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

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

### 2.1. Data Model Overview

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

#### 2.1.1. Features

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

Features:  
+-- https-initiate  
+-- http-listen  
+-- https-listen

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

#### 2.1.2. Groupings

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

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

Each of these groupings are presented in the following subsections.

##### 2.1.2.1. The "restconf-client-grouping" Grouping

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

```
grouping restconf-client-grouping ---> <empty>
```

Comments:

- \* This grouping does not define any nodes, but is maintained so that downstream modules can augment nodes into it if needed.
- \* The "restconf-client-grouping" defines, if it can be called that, the configuration for just "RESTCONF" part of a protocol stack. It does not, for instance, define any configuration for the "TCP", "TLS", or "HTTP" protocol layers (for that, see Section 2.1.2.2 and Section 2.1.2.3).

##### 2.1.2.2. The "restconf-client-initiate-stack-grouping" Grouping

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

```
grouping restconf-client-initiate-stack-grouping
  +-- (transport)
    +--:(https) {https-initiate}?
      +-- https
        +-- tcp-client-parameters
          | +---u tcpc:tcp-client-grouping
        +-- tls-client-parameters
          | +---u tlsc:tls-client-grouping
        +-- http-client-parameters
          | +---u httpc:http-client-grouping
        +-- restconf-client-parameters
          +---u rcc:restconf-client-grouping
```

Comments:

- \* The "restconf-client-initiate-stack-grouping" defines the configuration for a full RESTCONF protocol stack, for RESTCONF clients that initiate connections to RESTCONF servers, as opposed to receiving call-home [RFC8071] connections.
- \* The "transport" choice node enables transport options to be configured. This document only defines an "https" option, but other options MAY be augmented in.
- \* For the referenced grouping statement(s):
  - The "tcp-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
  - The "tls-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tls-client-server].
  - The "http-client-grouping" grouping is discussed in Section 2.1.2.2 of [I-D.ietf-netconf-http-client-server].
  - The "restconf-client-grouping" grouping is discussed in Section 2.1.2.1 in this document.

#### 2.1.2.3. The "restconf-client-listen-stack-grouping" Grouping

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

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

Comments:

- \* The "restconf-client-listen-stack-grouping" defines the configuration for a full RESTCONF protocol stack, for RESTCONF clients that receive call-home [RFC8071] connections from RESTCONF servers.
- \* The "transport" choice node enables both the HTTP and HTTPS transports to be configured, with each option enabled by a "feature" statement. Note that RESTCONF requires HTTPS, the HTTP option is provided to support cases where a TLS-terminator is deployed in front of the RESTCONF-client.
- \* For the referenced grouping statement(s):
  - The "tcp-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
  - The "tls-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tls-client-server].
  - The "http-client-grouping" grouping is discussed in Section 2.1.2.2 of [I-D.ietf-netconf-http-client-server].
  - The "restconf-client-grouping" grouping is discussed in Section 2.1.2.1 in this document.

#### 2.1.2.4. The "restconf-client-app-grouping" Grouping

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

```

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

Comments:

- \* The "restconf-client-app-grouping" defines the configuration for a RESTCONF client that supports both initiating connections to RESTCONF servers as well as receiving call-home connections from RESTCONF servers.
- \* Both the "initiate" and "listen" subtrees must be enabled by "feature" statements.
- \* For the referenced grouping statement(s):
  - The "restconf-client-initiate-stack-grouping" grouping is discussed in Section 2.1.2.2 in this document.
  - The "restconf-client-listen-stack-grouping" grouping is discussed in Section 2.1.2.3 in this document.

### 2.1.3. Protocol-accessible Nodes

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

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

Comments:

- \* Protocol-accessible nodes are those nodes that are accessible when the module is "implemented", as described in Section 5.6.5 of [RFC7950].
- \* For the "ietf-restconf-client" module, the protocol-accessible nodes are an instance of the "restconf-client-app-grouping" discussed in Section 2.1.2.4 grouping.
- \* The reason for why "restconf-client-app-grouping" exists separate from the protocol-accessible nodes definition is so as to enable instances of restconf-client-app-grouping to be instantiated in other locations, as may be needed or desired by some modules.

### 2.2. Example Usage

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

This example is consistent with the examples presented in Section 2.2 of [I-D.ietf-netconf-trust-anchors] and Section 2.2 of [I-D.ietf-netconf-keystore].

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

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

```
  <!-- RESTCONF servers to initiate connections to -->
  <initiate>
    <restconf-server>
      <name>corp-fw1</name>
      <endpoints>
        <endpoint>
          <name>corp-fw1.example.com</name>
          <https>
            <tcp-client-parameters>
              <remote-address>corp-fw1.example.com</remote-address>
```

```

        <keepalives>
          <idle-time>15</idle-time>
          <max-probes>3</max-probes>
          <probe-interval>30</probe-interval>
        </keepalives>
      </tcp-client-parameters>
      <tls-client-parameters>
        <client-identity>
          <certificate>
            <keystore-reference>
              <asymmetric-key>rsa-asymmetric-key</asymmetric-k\
ey>
              <certificate>ex-rsa-cert</certificate>
            </keystore-reference>
          </certificate>
        </client-identity>
        <server-authentication>
          <ca-certs>
            <truststore-reference>trusted-server-ca-certs</tru\
ststore-reference>
          </ca-certs>
          <ee-certs>
            <truststore-reference>trusted-server-ee-certs</tru\
ststore-reference>
          </ee-certs>
        </server-authentication>
      </keepalives>
      <test-peer-aliveness>
        <max-wait>30</max-wait>
        <max-attempts>3</max-attempts>
      </test-peer-aliveness>
    </keepalives>
  </tls-client-parameters>
  <http-client-parameters>
    <client-identity>
      <basic>
        <user-id>bob</user-id>
        <cleartext-password>secret</cleartext-password>
      </basic>
    </client-identity>
  </http-client-parameters>
</https>
</endpoint>
<endpoint>
  <name>corp-fw2.example.com</name>
  <https>
    <tcp-client-parameters>
      <remote-address>corp-fw2.example.com</remote-address>

```

```

        <keepalives>
          <idle-time>15</idle-time>
          <max-probes>3</max-probes>
          <probe-interval>30</probe-interval>
        </keepalives>
      </tcp-client-parameters>
      <tls-client-parameters>
        <client-identity>
          <certificate>
            <keystore-reference>
              <asymmetric-key>rsa-asymmetric-key</asymmetric-k\
ey>
              <certificate>ex-rsa-cert</certificate>
            </keystore-reference>
          </certificate>
        </client-identity>
        <server-authentication>
          <ca-certs>
            <truststore-reference>trusted-server-ca-certs</tru\
ststore-reference>
          </ca-certs>
          <ee-certs>
            <truststore-reference>trusted-server-ee-certs</tru\
ststore-reference>
          </ee-certs>
        </server-authentication>
      </keepalives>
      <test-peer-aliveness>
        <max-wait>30</max-wait>
        <max-attempts>3</max-attempts>
      </test-peer-aliveness>
    </keepalives>
  </tls-client-parameters>
  <http-client-parameters>
    <client-identity>
      <basic>
        <user-id>bob</user-id>
        <cleartext-password>secret</cleartext-password>
      </basic>
    </client-identity>
  </http-client-parameters>
</https>
</endpoint>
</endpoints>
<connection-type>
  <persistent/>
</connection-type>
</restconf-server>

```



```

</initiate>

<!-- endpoints to listen for RESTCONF Call Home connections on -->
<listen>
  <endpoint>
    <name>Intranet-facing listener</name>
    <https>
      <tcp-server-parameters>
        <local-address>11.22.33.44</local-address>
      </tcp-server-parameters>
      <tls-client-parameters>
        <client-identity>
          <certificate>
            <keystore-reference>
              <asymmetric-key>rsa-asymmetric-key</asymmetric-key>
              <certificate>ex-rsa-cert</certificate>
            </keystore-reference>
          </certificate>
        </client-identity>
        <server-authentication>
          <ca-certs>
            <truststore-reference>trusted-server-ca-certs</trustst\
ore-reference>
          </ca-certs>
          <ee-certs>
            <truststore-reference>trusted-server-ee-certs</trustst\
ore-reference>
          </ee-certs>
        </server-authentication>
        <keepalives>
          <peer-allowed-to-send/>
        </keepalives>
      </tls-client-parameters>
      <http-client-parameters>
        <client-identity>
          <basic>
            <user-id>bob</user-id>
            <cleartext-password>secret</cleartext-password>
          </basic>
        </client-identity>
      </http-client-parameters>
    </https>
  </endpoint>
</listen>
</restconf-client>

```

### 2.3. YANG Module

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

<CODE BEGINS> file "ietf-restconf-client@2022-03-07.yang"

```
module ietf-restconf-client {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-restconf-client";
  prefix rcc;

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

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

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

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

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

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

  contact
```

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

description

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

```
Copyright (c) 2021 IETF Trust and the persons identified
as authors of the code. All rights reserved.
```

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

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

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

```
revision 2022-03-07 {
  description
    "Initial version";
  reference
    "RFC IIII: RESTCONF Client and Server Models";
}
```

// Features

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

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

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

// Groupings

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

    This grouping currently does not define any nodes.";
}

grouping restconf-client-initiate-stack-grouping {
  description
    "A reusable grouping for configuring a RESTCONF client
    'initiate' protocol stack for a single connection.";

  choice transport {
    mandatory true;
    description
      "Selects between available transports. This is a
      'choice' statement so as to support additional
      transport options to be augmented in.";
    case https {
      if-feature "https-initiate";
      container https {
        must 'tls-client-parameters/client-identity
        or http-client-parameters/client-identity';
        description

```

```
    "Specifies HTTPS-specific transport
    configuration.";
  container tcp-client-parameters {
    description
      "A wrapper around the TCP client parameters
      to avoid name collisions.";
    uses tcpc:tcp-client-grouping {
      refine "remote-port" {
        default "443";
        description
          "The RESTCONF client will attempt to
          connect to the IANA-assigned well-known
          port value for 'https' (443) if no value
          is specified.";
      }
    }
  }
  container tls-client-parameters {
    description
      "A wrapper around the TLS client parameters
      to avoid name collisions.";
    uses tlsc:tls-client-grouping;
  }
  container http-client-parameters {
    description
      "A wrapper around the HTTP client parameters
      to avoid name collisions.";
    uses httpc:http-client-grouping;
  }
  container restconf-client-parameters {
    description
      "A wrapper around the HTTP client parameters
      to avoid name collisions.";
    uses rcc:restconf-client-grouping;
  }
}
}
} // restconf-client-initiate-stack-grouping

grouping restconf-client-listen-stack-grouping {
  description
    "A reusable grouping for configuring a RESTCONF client
    'listen' protocol stack for a single connection. The
    'listen' stack supports call home connections, as
    described in RFC 8071";
  reference
    "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
}
```

```
choice transport {
  mandatory true;
  description
    "Selects between available transports. This is a
    'choice' statement so as to support additional
    transport options to be augmented in.";
  case http {
    if-feature "http-listen";
    container http {
      description
        "HTTP-specific listening configuration for inbound
        connections.

        This transport option is made available to support
        deployments where the TLS connections are terminated
        by another system (e.g., a load balancer) fronting
        the client.";
      container tcp-server-parameters {
        description
          "A wrapper around the TCP client parameters
          to avoid name collisions.";
        uses tcps:tcp-server-grouping {
          refine "local-port" {
            default "4336";
            description
              "The RESTCONF client will listen on the IANA-
              assigned well-known port for 'restconf-ch-tls'
              (4336) if no value is specified.";
          }
        }
      }
      container http-client-parameters {
        description
          "A wrapper around the HTTP client parameters
          to avoid name collisions.";
        uses httpc:http-client-grouping;
      }
      container restconf-client-parameters {
        description
          "A wrapper around the RESTCONF client parameters
          to avoid name collisions.";
        uses rcc:restconf-client-grouping;
      }
    }
  }
  case https {
    if-feature "https-listen";
    container https {
```

```
    must 'tls-client-parameters/client-identity
        or http-client-parameters/client-identity';
    description
        "HTTPS-specific listening configuration for inbound
        connections.";
    container tcp-server-parameters {
        description
            "A wrapper around the TCP client parameters
            to avoid name collisions.";
        uses tcps:tcp-server-grouping {
            refine "local-port" {
                default "4336";
                description
                    "The RESTCONF client will listen on the IANA-
                    assigned well-known port for 'restconf-ch-tls'
                    (4336) if no value is specified.";
            }
        }
    }
    container tls-client-parameters {
        description
            "A wrapper around the TLS client parameters
            to avoid name collisions.";
        uses tlsc:tls-client-grouping;
    }
    container http-client-parameters {
        description
            "A wrapper around the HTTP client parameters
            to avoid name collisions.";
        uses httpc:http-client-grouping;
    }
    container restconf-client-parameters {
        description
            "A wrapper around the RESTCONF client parameters
            to avoid name collisions.";
        uses rcc:restconf-client-grouping;
    }
}
}
} // restconf-client-listen-stack-grouping

grouping restconf-client-app-grouping {
    description
        "A reusable grouping for configuring a RESTCONF client
        application that supports both 'initiate' and 'listen'
        protocol stacks for a multiplicity of connections.";
    container initiate {
```

```
if-feature "https-initiate";
presence
  "Indicates that client-initiated connections have been
  configured. This statement is present so the mandatory
  descendant nodes do not imply that this node must be
  configured.";
description
  "Configures client initiating underlying TCP connections.";
list restconf-server {
  key "name";
  min-elements 1;
  description
    "List of RESTCONF servers the RESTCONF client is to
    maintain simultaneous connections with.";
  leaf name {
    type string;
    description
      "An arbitrary name for the RESTCONF server.";
  }
  container endpoints {
    description
      "Container for 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.";
      }
      uses restconf-client-initiate-stack-grouping;
    }
  }
}
container connection-type {
  description
    "Indicates the RESTCONF client's preference for how
    the RESTCONF connection is maintained.";
  choice connection-type {
    mandatory true;
    description
      "Selects between available connection types.";
    case persistent-connection {
      container persistent {
```



```
presence
  "Indicates that a persistent connection is to be
  maintained.";
description
  "Maintain a persistent connection to the
  RESTCONF server. If the connection goes down,
  immediately start trying to reconnect to the
  RESTCONF server, using the reconnection strategy.

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

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

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

      In the case that the previous connection is
      still active, establishing a new connection
      is NOT RECOMMENDED.";
    leaf period {
      type uint16;
      units "minutes";
      default "60";
      description
        "Duration of time between periodic
        connections.";
    }
    leaf anchor-time {
      type yang:date-and-time {
        // constrained to minute-level granularity
        pattern '\d{4}-\d{2}-\d{2}T\d{2}:\d{2}'
          + '(Z|[\+|-]\d{2}:\d{2})';
      }
    }
  }
}
```

```
description
    "Designates a timestamp before or after which
    a series of periodic connections are
    determined. The periodic connections occur
    at a whole multiple interval from the anchor
    time. For example, for an anchor time is 15
    minutes past midnight and a period interval
    of 24 hours, then a periodic connection will
    occur 15 minutes past midnight everyday.";
}
leaf idle-timeout {
    type uint16;
    units "seconds";
    default "120"; // two minutes
    description
        "Specifies the maximum number of seconds
        that the underlying TCP session may remain
        idle. A TCP session will be dropped if it
        is idle for an interval longer than this
        number of seconds. If set to zero, then the
        RESTCONF client will never drop a session
        because it is idle.";
}
} // periodic-connection
} // connection-type
} // connection-type
container reconnect-strategy {
    description
        "The reconnection strategy directs how a RESTCONF
        client reconnects to a RESTCONF server, after
        discovering its connection to the server has
        dropped, even if due to a reboot. The RESTCONF
        client starts with the specified endpoint and
        tries to connect to it max-attempts times before
        trying the next endpoint in the list (round
        robin).";
    leaf start-with {
        type enumeration {
            enum first-listed {
                description
                    "Indicates that reconnections should start
                    with the first endpoint listed.";
            }
            enum last-connected {
                description
                    "Indicates that reconnections should start
                    with the endpoint last connected to. If
```

```
        no previous connection has ever been
        established, then the first endpoint
        configured is used.  RESTCONF clients
        SHOULD be able to remember the last
        endpoint connected to across reboots.";
    }
    enum random-selection {
        description
            "Indicates that reconnections should start with
            a random endpoint.";
    }
}
default "first-listed";
description
    "Specifies which of the RESTCONF server's
    endpoints the RESTCONF client should start
    with when trying to connect to the RESTCONF
    server.";
}
leaf max-attempts {
    type uint8 {
        range "1..max";
    }
    default "3";
    description
        "Specifies the number times the RESTCONF client
        tries to connect to a specific endpoint before
        moving on to the next endpoint in the list
        (round robin).";
}
}
} // initiate
container listen {
    if-feature "http-listen or https-listen";
    presence
        "Indicates that client-listening ports have been configured.
        This statement is present so the mandatory descendant nodes
        do not imply that this node must be configured.";
    description
        "Configures the client to accept call-home TCP connections.";
    leaf idle-timeout {
        type uint16;
        units "seconds";
        default "3600"; // one hour
        description
            "Specifies the maximum number of seconds that an
            underlying TCP session may remain idle. A TCP session
```

```
        will be dropped if it is idle for an interval longer
        then 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.";
        }
        uses restconf-client-listen-stack-grouping;
    }
}
} // restconf-client-app-grouping

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

<CODE ENDS>

### 3. The "ietf-restconf-server" Module

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

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

#### 3.1. Data Model Overview

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

### 3.1.1. Features

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

Features:

```
+-- http-listen
+-- https-listen
+-- https-call-home
```

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

### 3.1.2. Groupings

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

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

Each of these groupings are presented in the following subsections.

#### 3.1.2.1. The "restconf-server-grouping" Grouping

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

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

Comments:

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

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

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

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

```

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

```

Comments:

- \* The "restconf-server-listen-stack-grouping" defines the configuration for a full RESTCONF protocol stack for RESTCONF servers that listen for standard connections from RESTCONF clients, as opposed to initiating call-home [RFC8071] connections.
- \* The "transport" choice node enables both the HTTP and HTTPS transports to be configured, with each option enabled by a "feature" statement. The HTTP option is provided to support cases where a TLS-terminator is deployed in front of the RESTCONF-server.
- \* For the referenced grouping statement(s):

- The "tcp-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
- The "tls-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tls-client-server].
- The "http-server-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-http-client-server].
- The "restconf-server-grouping" is discussed in Section 3.1.2.1 of this document.

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

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

```

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

```

Comments:

- \* The "restconf-server-callhome-stack-grouping" defines the configuration for a full RESTCONF protocol stack, for RESTCONF servers that initiate call-home [RFC8071] connections to RESTCONF clients.
- \* The "transport" choice node enables transport options to be configured. This document only defines an "https" option, but other options MAY be augmented in.
- \* For the referenced grouping statement(s):
  - The "tcp-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
  - The "tls-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tls-client-server].
  - The "http-server-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-http-client-server].
  - The "restconf-server-grouping" is discussed in Section 3.1.2.1 of this document.

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

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

```

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

```

Comments:

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



### 3.1.3. Protocol-accessible Nodes

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

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

Comments:

- \* Protocol-accessible nodes are those nodes that are accessible when the module is "implemented", as described in Section 5.6.5 of [RFC7950].
- \* For the "ietf-restconf-server" module, the protocol-accessible nodes are an instance of the "restconf-server-app-grouping" discussed in Section 3.1.2.4 grouping.
- \* The reason for why "restconf-server-app-grouping" exists separate from the protocol-accessible nodes definition is so as to enable instances of restconf-server-app-grouping to be instantiated in other locations, as may be needed or desired by some modules.

### 3.2. Example Usage

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

This example is consistent with the examples presented in Section 2.2 of [I-D.ietf-netconf-trust-anchors] and Section 2.2 of [I-D.ietf-netconf-keystore].

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

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

```
  <!-- endpoints to listen for RESTCONF connections on -->
  <listen>
    <endpoint>
      <name>restconf/https</name>
      <https>
        <tcp-server-parameters>
          <local-address>11.22.33.44</local-address>
        </tcp-server-parameters>
        <tls-server-parameters>
```

```

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

<!-- call home to a RESTCONF client with two endpoints -->
<call-home>
  <restconf-client>
    <name>config-manager</name>
    <endpoints>

```

```

    <endpoint>
      <name>east-data-center</name>
      <https>
        <tcp-client-parameters>
          <remote-address>east.example.com</remote-address>
          <keepalives>
            <idle-time>15</idle-time>
            <max-probes>3</max-probes>
            <probe-interval>30</probe-interval>
          </keepalives>
        </tcp-client-parameters>
        <tls-server-parameters>
          <server-identity>
            <certificate>
              <keystore-reference>
                <asymmetric-key>rsa-asymmetric-key</asymmetric-k\
ey>
                <certificate>ex-rsa-cert</certificate>
              </keystore-reference>
            </certificate>
          </server-identity>
          <client-authentication>
            <ca-certs>
              <truststore-reference>trusted-client-ca-certs</tru\
ststore-reference>
            </ca-certs>
            <ee-certs>
              <truststore-reference>trusted-client-ee-certs</tru\
ststore-reference>
            </ee-certs>
          </client-authentication>
          <keepalives>
            <test-peer-aliveness>
              <max-wait>30</max-wait>
              <max-attempts>3</max-attempts>
            </test-peer-aliveness>
          </keepalives>
        </tls-server-parameters>
        <http-server-parameters>
          <server-name>foo.example.com</server-name>
        </http-server-parameters>
        <restconf-server-parameters>
          <client-identity-mappings>
            <cert-to-name>
              <id>1</id>
              <fingerprint>11:0A:05:11:00</fingerprint>
              <map-type>x509c2n:specified</map-type>
              <name>scooby-doo</name>
            </cert-to-name>
          </client-identity-mappings>
        </restconf-server-parameters>
      </https>
    </endpoint>
  </data>
</restconf>

```

```

        </cert-to-name>
        <cert-to-name>
          <id>2</id>
          <map-type>x509c2n:san-any</map-type>
        </cert-to-name>
      </client-identity-mappings>
    </restconf-server-parameters>
  </https>
</endpoint>
<endpoint>
  <name>west-data-center</name>
  <https>
    <tcp-client-parameters>
      <remote-address>west.example.com</remote-address>
      <keepalives>
        <idle-time>15</idle-time>
        <max-probes>3</max-probes>
        <probe-interval>30</probe-interval>
      </keepalives>
    </tcp-client-parameters>
    <tls-server-parameters>
      <server-identity>
        <certificate>
          <keystore-reference>
            <asymmetric-key>rsa-asymmetric-key</asymmetric-k\
ey>
            <certificate>ex-rsa-cert</certificate>
          </keystore-reference>
        </certificate>
      </server-identity>
      <client-authentication>
        <ca-certs>
          <truststore-reference>trusted-client-ca-certs</tru\
ststore-reference>
        </ca-certs>
        <ee-certs>
          <truststore-reference>trusted-client-ee-certs</tru\
ststore-reference>
        </ee-certs>
      </client-authentication>
      <keepalives>
        <test-peer-aliveness>
          <max-wait>30</max-wait>
          <max-attempts>3</max-attempts>
        </test-peer-aliveness>
      </keepalives>
    </tls-server-parameters>
  </https>
</http-server-parameters>

```

```

        <server-name>foo.example.com</server-name>
      </http-server-parameters>
    <restconf-server-parameters>
      <client-identity-mappings>
        <cert-to-name>
          <id>1</id>
          <fingerprint>11:0A:05:11:00</fingerprint>
          <map-type>x509c2n:specified</map-type>
          <name>scooby-doo</name>
        </cert-to-name>
        <cert-to-name>
          <id>2</id>
          <map-type>x509c2n:san-any</map-type>
        </cert-to-name>
      </client-identity-mappings>
    </restconf-server-parameters>
  </https>
</endpoint>
</endpoints>
<connection-type>
  <periodic>
    <idle-timeout>300</idle-timeout>
    <period>60</period>
  </periodic>
</connection-type>
<reconnect-strategy>
  <start-with>last-connected</start-with>
  <max-attempts>3</max-attempts>
</reconnect-strategy>
</restconf-client>
</call-home>
</restconf-server>

```

### 3.3. YANG Module

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

```
<CODE BEGINS> file "ietf-restconf-server@2022-03-07.yang"
```

```

module ietf-restconf-server {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-restconf-server";
  prefix rcs;

  import ietf-yang-types {

```

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

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

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

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

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

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

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

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

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

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) 2021 IETF Trust and the persons identified  
as authors of the code. All rights reserved.

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

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

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

revision 2022-03-07 {

description

"Initial version";

reference

"RFC IIII: RESTCONF Client and Server Models";

}

// Features

feature http-listen {

description

"The 'http-listen' feature indicates that the RESTCONF server  
supports opening a port to listen for incoming RESTCONF over  
TPC client connections, whereby the TLS connections are  
terminated by an external system.";

reference

"RFC 8040: RESTCONF Protocol";

}

```
feature https-listen {
  description
    "The 'https-listen' feature indicates that the RESTCONF server
    supports opening a port to listen for incoming RESTCONF over
    TLS client connections, whereby the TLS connections are
    terminated by the server itself.";
  reference
    "RFC 8040: RESTCONF Protocol";
}

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

// Groupings

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

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

  container client-identity-mappings {
    description
      "Specifies mappings through which RESTCONF client X.509
      certificates are used to determine a RESTCONF username.
      If no matching and valid cert-to-name list entry can be
      found, then the RESTCONF server MUST close the connection,
      and MUST NOT accept RESTCONF messages over it.";
    reference
      "RFC 7407: A YANG Data Model for SNMP Configuration.";
    uses x509c2n:cert-to-name {
      refine "cert-to-name/fingerprint" {
        mandatory false;
        description
```



```
        "A 'fingerprint' value does not need to be specified
        when the 'cert-to-name' mapping is independent of
        fingerprint matching. A 'cert-to-name' having no
        fingerprint value will match any client certificate
        and therefore should only be present at the end of
        the user-ordered 'cert-to-name' list.";
    }
}
}

grouping restconf-server-listen-stack-grouping {
  description
    "A reusable grouping for configuring a RESTCONF server
    'listen' protocol stack for a single connection.";
  choice transport {
    mandatory true;
    description
      "Selects between available transports. This is a
      'choice' statement so as to support additional
      transport options to be augmented in.";
    case http {
      if-feature "http-listen";
      container http {
        description
          "Configures RESTCONF server stack assuming that
          TLS-termination is handled externally.";
        container external-endpoint {
          presence
            "Identifies that an external endpoint has been
            configured. This statement is present so the
            mandatory descendant nodes do not imply that
            this node must be configured.";
          description
            "Identifies contact information for the external
            system that terminates connections before passing
            them thru to this server (e.g., a network address
            translator or a load balancer). These values have
            no effect on the local operation of this server,
            but may be used by the application when needing to
            inform other systems how to contact this server.";
          leaf address {
            type inet:ip-address;
            mandatory true;
            description
              "The IP address or hostname of the external system
              that terminates incoming RESTCONF client
              connections before forwarding them to this
```

```
        server.";
    }
    leaf port {
        type inet:port-number;
        default "443";
        description
            "The port number that the external system listens
            on for incoming RESTCONF client connections that
            are forwarded to this server. The default HTTPS
            port (443) is used, as expected for a RESTCONF
            connection.";
    }
}
container tcp-server-parameters {
    description
        "A wrapper around the TCP server parameters
        to avoid name collisions.";
    uses tcps:tcp-server-grouping {
        refine "local-port" {
            default "80";
            description
                "The RESTCONF server will listen on the IANA-
                assigned well-known port value for 'http'
                (80) if no value is specified.";
        }
    }
}
container http-server-parameters {
    description
        "A wrapper around the HTTP server parameters
        to avoid name collisions.";
    uses https:http-server-grouping;
}
container restconf-server-parameters {
    description
        "A wrapper around the RESTCONF server parameters
        to avoid name collisions.";
    uses rcs:restconf-server-grouping;
}
}
}
case https {
    if-feature "https-listen";
    container https {
        description
            "Configures RESTCONF server stack assuming that
            TLS-termination is handled internally.";
        container tcp-server-parameters {
```

```
    description
      "A wrapper around the TCP server parameters
      to avoid name collisions.";
    uses tcps:tcp-server-grouping {
      refine "local-port" {
        default "443";
        description
          "The RESTCONF server will listen on the IANA-
          assigned well-known port value for 'https'
          (443) if no value is specified.";
      }
    }
  }
  container tls-server-parameters {
    description
      "A wrapper around the TLS server parameters
      to avoid name collisions.";
    uses tlss:tls-server-grouping;
  }
  container http-server-parameters {
    description
      "A wrapper around the HTTP server parameters
      to avoid name collisions.";
    uses https:http-server-grouping;
  }
  container restconf-server-parameters {
    description
      "A wrapper around the RESTCONF server parameters
      to avoid name collisions.";
    uses rcs:restconf-server-grouping;
  }
}

grouping restconf-server-callhome-stack-grouping {
  description
    "A reusable grouping for configuring a RESTCONF server
    'call-home' protocol stack, for a single connection.";
  choice transport {
    mandatory true;
    description
      "Selects between available transports. This is a
      'choice' statement so as to support additional
      transport options to be augmented in.";
    case https {
      if-feature "https-listen";
    }
  }
}
```

```
container https {
  description
    "Configures RESTCONF server stack assuming that
    TLS-termination is handled internally.";
  container tcp-client-parameters {
    description
      "A wrapper around the TCP client parameters
      to avoid name collisions.";
    uses tcpc:tcp-client-grouping {
      refine "remote-port" {
        default "4336";
        description
          "The RESTCONF server will attempt to
          connect to the IANA-assigned well-known
          port for 'restconf-ch-tls' (4336) if no
          value is specified.";
      }
    }
  }
  container tls-server-parameters {
    description
      "A wrapper around the TLS server parameters
      to avoid name collisions.";
    uses tlss:tls-server-grouping;
  }
  container http-server-parameters {
    description
      "A wrapper around the HTTP server parameters
      to avoid name collisions.";
    uses https:http-server-grouping;
  }
  container restconf-server-parameters {
    description
      "A wrapper around the RESTCONF server parameters
      to avoid name collisions.";
    uses rcs:restconf-server-grouping;
  }
}

grouping restconf-server-app-grouping {
  description
    "A reusable grouping for configuring a RESTCONF server
    application that supports both 'listen' and 'call-home'
    protocol stacks for a multiplicity of connections.";
  container listen {
```

```
if-feature "http-listen or https-listen";
presence
  "Identifies that the server has been configured to
  listen for incoming client connections. This statement
  is present so the mandatory descendant nodes do not
  imply that this node must be configured.";
description
  "Configures the RESTCONF server to listen for RESTCONF
  client connections.";
list endpoint {
  key "name";
  min-elements 1;
  description
    "List of endpoints to listen for RESTCONF connections.";
  leaf name {
    type string;
    description
      "An arbitrary name for the RESTCONF listen endpoint.";
  }
  uses restconf-server-listen-stack-grouping;
}
}
container call-home {
  if-feature "https-call-home";
  presence
    "Identifies that the server has been configured to initiate
    call home connections. This statement is present so the
    mandatory descendant nodes do not imply that this node
    must be configured.";
  description
    "Configures the RESTCONF server to initiate the underlying
    transport connection to RESTCONF clients.";
  list restconf-client {
    key "name";
    min-elements 1;
    description
      "List of RESTCONF clients the RESTCONF server is to
      maintain simultaneous call-home connections with.";
    leaf name {
      type string;
      description
        "An arbitrary name for the remote RESTCONF client.";
    }
  }
  container endpoints {
    description
      "Container for the list of endpoints.";
    list endpoint {
      key "name";
```

```
    min-elements 1;
    ordered-by user;
    description
      "User-ordered list of endpoints for this RESTCONF
       client. Defining more than one enables high-
       availability.";
    leaf name {
      type string;
      description
        "An arbitrary name for this endpoint.";
    }
    uses restconf-server-callhome-stack-grouping;
  }
}

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

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

           This connection type increases resource
```

utilization, albeit with increased delay in RESTCONF client to RESTCONF client interactions.

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

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

```
leaf period {
  type uint16;
  units "minutes";
  default "60";
  description
    "Duration of time between periodic connections.";
}
leaf anchor-time {
  type yang:date-and-time {
    // constrained to minute-level granularity
    pattern '\d{4}-\d{2}-\d{2}T\d{2}:\d{2}'
      + '(Z|[\+|-]\d{2}:\d{2})';
  }
  description
    "Designates a timestamp before or after which a
    series of periodic connections are determined.
    The periodic connections occur at a whole
    multiple interval from the anchor time. For
    example, for an anchor time is 15 minutes past
    midnight and a period interval of 24 hours, then
    a periodic connection will occur 15 minutes past
    midnight everyday.";
}
leaf idle-timeout {
  type uint16;
  units "seconds";
  default "120"; // two minutes
  description
    "Specifies the maximum number of seconds that
    the underlying TCP session may remain idle.
    A TCP session will be dropped if it is idle
    for an interval longer than this number of
    seconds. If set to zero, then the server
```

```
        will never drop a session because it is idle.";
    }
}
}
}
container reconnect-strategy {
  description
    "The reconnection strategy directs how a RESTCONF server
    reconnects to a RESTCONF client after discovering its
    connection to the client has dropped, even if due to a
    reboot. The RESTCONF server starts with the specified
    endpoint and tries to connect to it max-attempts times
    before trying the next endpoint in the list (round
    robin).";
  leaf start-with {
    type enumeration {
      enum first-listed {
        description
          "Indicates that reconnections should start with
          the first endpoint listed.";
      }
      enum last-connected {
        description
          "Indicates that reconnections should start with
          the endpoint last connected to. If no previous
          connection has ever been established, then the
          first endpoint configured is used. RESTCONF
          servers SHOULD be able to remember the last
          endpoint connected to across reboots.";
      }
      enum random-selection {
        description
          "Indicates that reconnections should start with
          a random endpoint.";
      }
    }
    default "first-listed";
    description
      "Specifies which of the RESTCONF client's endpoints
      the RESTCONF server should start with when trying
      to connect to the RESTCONF client.";
  }
  leaf max-attempts {
    type uint8 {
      range "1..max";
    }
    default "3";
  }
}
```



```
        description
          "Specifies the number times the RESTCONF server tries
           to connect to a specific endpoint before moving on to
           the next endpoint in the list (round robin).";
      }
    } // restconf-client
  } // call-home
} // restconf-server-app-grouping

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

<CODE ENDS>

## 4. Security Considerations

### 4.1. The "ietf-restconf-client" YANG Module

The "ietf-restconf-client" YANG module defines data nodes that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. 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.

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

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

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

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

#### 4.2. The "ietf-restconf-server" YANG Module

The "ietf-restconf-server" YANG module defines data nodes that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. 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.

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

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

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

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

### 5. IANA Considerations

#### 5.1. The "IETF XML" Registry

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

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

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

## 5.2. The "YANG Module Names" Registry

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

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

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

## 6. References

### 6.1. Normative References

- [I-D.ietf-netconf-http-client-server]  
Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-08, 14 December 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-http-client-server-08>>.
- [I-D.ietf-netconf-keystore]  
Watsen, K., "A YANG Data Model for a Keystore", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-23, 14 December 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-keystore-23>>.
- [I-D.ietf-netconf-tcp-client-server]  
Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tcp-client-server-11, 14 December 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tcp-client-server-11>>.
- [I-D.ietf-netconf-tls-client-server]  
Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-26, 14 December 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tls-client-server-26>>.

- [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-crypto-types]  
Watsen, K., "YANG Data Types and Groupings for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-21, 14 September 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-crypto-types-21>>.

- [I-D.ietf-netconf-netconf-client-server]  
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-24, 14 December 2021,  
<<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-netconf-client-server-24>>.
- [I-D.ietf-netconf-restconf-client-server]  
Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-24, 14 December 2021,  
<<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-restconf-client-server-24>>.
- [I-D.ietf-netconf-ssh-client-server]  
Watsen, K., "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-ssh-client-server-26, 14 December 2021,  
<<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-ssh-client-server-26>>.
- [I-D.ietf-netconf-trust-anchors]  
Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-16, 14 December 2021,  
<<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-trust-anchors-16>>.
- [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>>.

[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>>.

[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

This section is to be removed before publishing as an RFC.

### A.1. 00 to 01

- \* Renamed "keychain" to "keystore".

### A.2. 01 to 02

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

### A.3. 02 to 03

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

### A.4. 03 to 04

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

## A.5. 04 to 05

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

## A.6. 05 to 06

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

## A.7. 06 to 07

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

## A.8. 07 to 08

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

## A.9. 08 to 09

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

## A.10. 09 to 10

- \* Reformatted YANG modules.

## A.11. 10 to 11

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

## A.12. 11 to 12

- \* Removed the 'must' statement limiting keepalives in periodic connections.
- \* Updated models and examples to reflect removal of the "demux" containers in the imported models.
- \* Updated the "periodic-connection" description statements to better describe behavior when connections are not closed gracefully.
- \* Updated text to better reference where certain examples come from (e.g., which Section in which draft).
- \* In the server model, commented out the "must 'pinned-ca-certs or pinned-client-certs'" statement to reflect change made in the TLS draft whereby the trust anchors MAY be defined externally.
- \* Replaced the 'listen', 'initiate', and 'call-home' features with boolean expressions.

## A.13. 12 to 13

- \* Updated to reflect changes in trust-anchors drafts (e.g., s/trust-anchors/truststore/g + s/pinned.//)
- \* In ietf-restconf-server, Added 'http-listen' (not https-listen) choice, to support case when server is behind a TLS-terminator.



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

## A.14. 13 to 14

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

## A.15. 14 to 15

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

## A.16. 15 to 16

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

## A.17. 16 to 17

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

## A.18. 17 to 18

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

## A.19. 18 to 19

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

## A.20. 19 to 20

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

## A.21. 20 to 21

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

## A.22. 21 to 22

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

## A.23. 22 to 23

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

## A.24. 23 to 24

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

- \* Minor editorial nits

A.25. 24 to 25

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

#### Acknowledgements

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

#### Author's Address

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

NETCONF  
Internet-Draft  
Intended status: Standards Track  
Expires: September 14, 2017

E. Voit  
A. Gonzalez Prieto  
A. Tripathy  
E. Nilsen-Nygaard  
Cisco Systems  
A. Clemm  
Huawei  
A. Bierman  
YumaWorks  
March 13, 2017

Restconf and HTTP Transport for Event Notifications  
draft-ietf-netconf-restconf-notif-02

Abstract

This document defines Restconf, HTTP2, and HTTP1.1 bindings for the transport of Subscription requests and corresponding push updates. Being subscribed may be either Event Notifications or objects or subtrees of YANG Datastores.

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 <http://datatracker.ietf.org/drafts/current/>.

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

This Internet-Draft will expire on September 14, 2017.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Terminology . . . . .	3
3. Solution . . . . .	3
3.1. Dynamic YANG Subscription with RESTCONF control . . . . .	3
3.2. Subscription Multiplexing . . . . .	6
4. Encoded Subscription and Event Notification Examples . . . . .	7
4.1. Restconf Subscription and Events over HTTP1.1 . . . . .	7
4.2. Event Notification over HTTP2 . . . . .	12
5. Security Considerations . . . . .	12
6. Acknowledgments . . . . .	13
7. References . . . . .	13
7.1. Normative References . . . . .	13
7.2. Informative References . . . . .	14
Appendix A. End-to-End Deployment Guidance . . . . .	14
A.1. Call Home . . . . .	14
A.2. TLS Heartbeat . . . . .	15
Appendix B. Issues being worked and resolved . . . . .	15
B.1. Unresolved Issues . . . . .	15
Appendix C. Changes between revisions . . . . .	15
Authors' Addresses . . . . .	16

## 1. Introduction

Mechanisms to support Event subscription and push are defined in [sn]. Enhancements to [sn] which enable YANG Datastore subscription and push are defined in [yang-push]. This document provides a transport specification for these protocols over Restconf and HTTP. Driving these requirements is [RFC7923].

The streaming of Subscription Event Notifications has synergies with HTTP2 streams. Benefits which can be realized when transporting events directly HTTP2 [RFC7540] include:

- o Elimination of head-of-line blocking
- o Weighting and proportional dequeuing of Events from different subscriptions
- o Explicit precedence in subscriptions so that events from one subscription must be sent before another dequeues

## 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 [sn]: Configured Subscription, Dynamic Subscription, Event Notification, Publisher, Receiver, Subscriber, Subscription.

## 3. Solution

Event subscription is defined in [sn], YANG Datastore subscription is defined in [yang-push]. This section specifies transport mechanisms applicable to both.

### 3.1. Dynamic YANG Subscription with RESTCONF control

Dynamic Subscriptions for both [sn] and its [yang-push] augmentations are configured and managed via signaling messages transported over [RFC8040]. These interactions will be accomplished via a Restconf POST into RPCs located on the Publisher. HTTP responses codes will indicate the results of the interaction with the Publisher. An HTTP status code of 200 is the proper response to a successful <establish-subscription> RPC call. The successful <establish-subscription> will result in a HTTP message with returned subscription URI on a logically separate mechanism than was used for the original Restconf POST. This mechanism is via a parallel TCP connection in the case of HTTP 1.x, or in the case of HTTP2 via a separate HTTP stream within the HTTP connection. When a being returned by the Publisher, failure will be indicated by error codes transported in payload.

Once established, the resulting stream of Event Notifications are then delivered via SSE for HTTP1.1 and via HTTP Data for HTTP2.

#### 3.1.1. Call Flow for HTTP2

Requests to [yang-push] augmented RPCs are sent on one or more HTTP2 streams indicated by (a) in Figure 2. Event Notifications related to a single subscription are pushed on a unique logical channel (b). In the case below, a newly established subscription has its events pushed over HTTP2 stream (7).

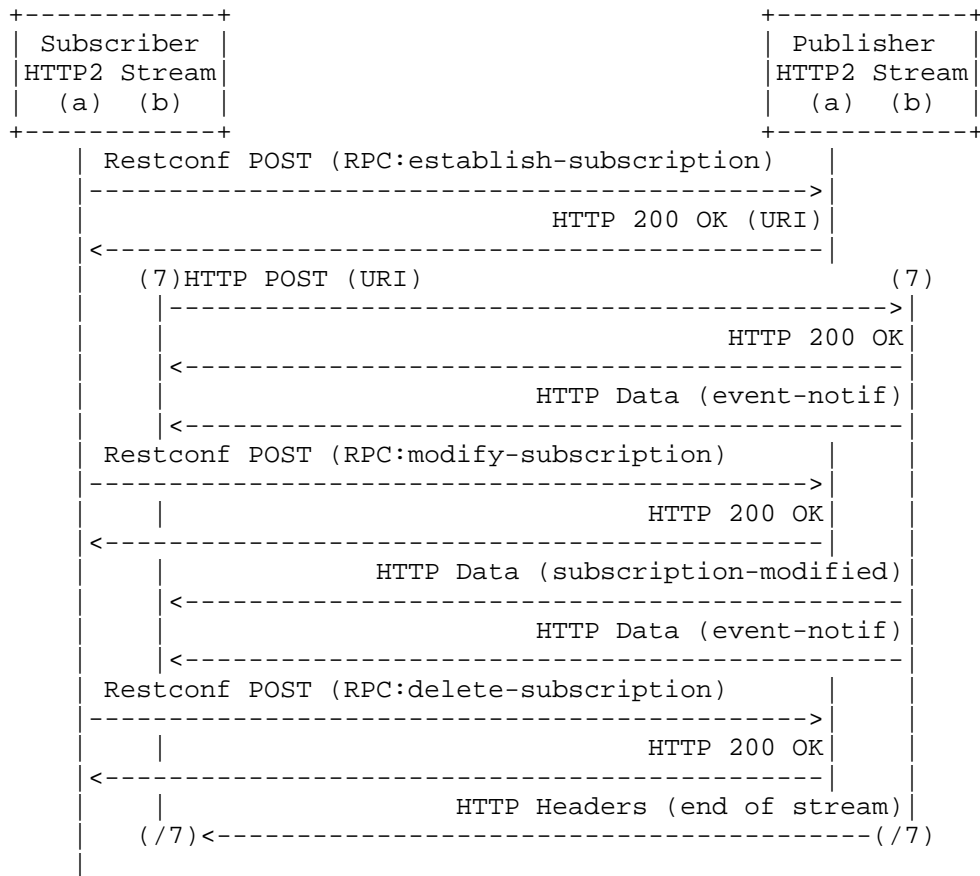


Figure 1: Dynamic with HTTP2

## 3.1.2. Call flow for HTTP1.1

Requests to [yang-push] RPCs are sent on the TCP connection indicated by (a). Event Notifications are pushed on a separate connection (b). This connection (b) will be used for all Event Notifications across all subscriptions.

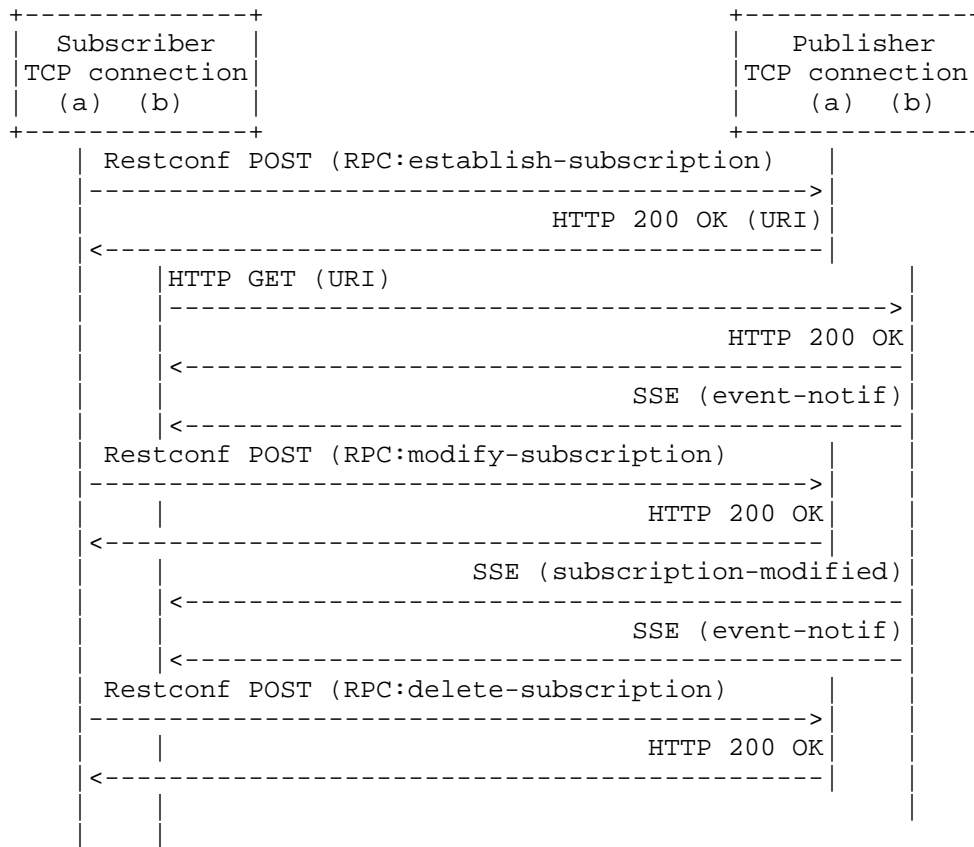


Figure 2: Dynamic with HTTP1.1

### 3.1.3. Configured Subscription over HTTP2

With a Configured Subscription, all information needed to establish a secure relationship with that Receiver is available on the Publisher. With this information, the Publisher will establish a secure transport connection with the Receiver and then begin pushing the Event Notifications to the Receiver. Since Restconf might not exist on the Receiver, it is not desirable to require that such Event Notifications be pushed with any dependency on Restconf. Nor is there value which Restconf provides on top of HTTP. Therefore in place of Restconf, a TLS secured HTTP2 Client connection must be established with an HTTP2 Server located on the Receiver. Event Notifications will then be sent as part of an extended HTTP POST to the Receiver.



POST messages will be addressed to HTTP augmentation code on the Receiver capable of accepting and responding to Event Notifications. The first POST message must be a subscription-started notification. Push update notifications must not be sent until the receipt of an HTTP 200 OK for this initial notification. The 200 OK will indicate that the Receiver is ready for Event Notifications. At this point a Subscription must be allocated its own HTTP2 stream. Figure 4 depicts this message flow.

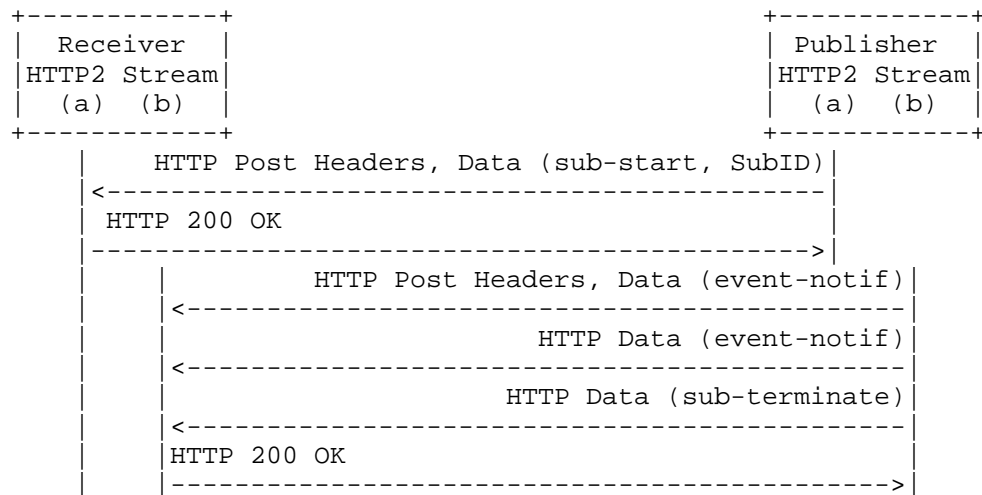


Figure 3: Configured over HTTP2

As the HTTP2 transport is available to the Receiver, the Publisher should:

- o take any subscription-priority and copy it into the HTTP2 stream priority, and
- o take a subscription-dependency if it has been provided and map the HTTP2 stream for the parent subscription into the HTTP2 stream dependency.

### 3.2. Subscription Multiplexing

It is possible that updates might be delivered in a different sequence than generated. Reasons for this might include (but are not limited to):

- o replay of pushed updates

- o temporary loss of transport connectivity, with update buffering and different dequeuing priorities per Subscription
- o population, marshalling and bundling of independent Subscription Updates, and

Therefore each Event Notification will include a timestamp to ensure that a Receiver understands the time when a that update was generated. Use of this timestamp can give an indication of the state of objects at a Publisher when state-entangled information is received across different subscriptions. The use of the latest Event Notification timestamp for a particular object update can introduce errors. So when state-entangled updates have inconsistent object values and temporally close timestamps, a Receiver might consider performing a GET to validate the current state of a Publisher.

#### 4. Encoded Subscription and Event Notification Examples

Transported updates will contain context data for one or more Event Notifications. Each transported Event Notification will contain several parameters:

##### 4.1. Restconf Subscription and Events over HTTP1.1

Subscribers can dynamically learn whether a RESTCONF server supports various types of Event or Yang datastore subscription capabilities. This is done by issuing an HTTP request OPTIONS, HEAD, or GET on the stream. Some examples building upon the Call flow for HTTP1.1 from Section 3.2.2 are:

```
GET /restconf/data/ietf-restconf-monitoring:restconf-state/  
    streams/stream=yang-push HTTP/1.1  
Host: example.com  
Accept: application/yang.data+xml
```

If the server supports it, it may respond

```
HTTP/1.1 200 OK
Content-Type: application/yang.api+xml
<stream xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf-monitoring">
  <name>yang-push</name>
  <description>Yang push stream</description>
  <access>
    <encoding>xml</encoding>
    <location>https://example.com/streams/yang-push-xml
  </access>
  <access>
    <encoding>json</encoding>
    <location>https://example.com/streams/yang-push-json
  </access>
</stream>
```

If the server does not support any form of subscription, it may respond

```
HTTP/1.1 404 Not Found
Date: Mon, 25 Apr 2012 11:10:30 GMT
Server: example-server
```

Subscribers can determine the URL to receive updates by sending an HTTP GET as a request for the "location" leaf with the stream list entry. The stream to use for may be selected from the Event Stream list provided in the capabilities exchange. Note that different encodings are supporting using different Event Stream locations. For example, the Subscriber might send the following request:

```
GET /restconf/data/ietf-restconf-monitoring:restconf-state/
    streams/stream=yang-push/access=xml/location HTTP/1.1
Host: example.com
Accept: application/yang.data+xml
```

The Publisher might send the following response:

```
HTTP/1.1 200 OK
Content-Type: application/yang.api+xml
  <location
    xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf-monitoring">
    https://example.com/streams/yang-push-xml
  </location>
```

To subscribe and start receiving updates, the subscriber can then send an HTTP GET request for the URL returned by the Publisher in the request above. The accept header must be "text/event-stream". The

Publisher uses the Server Sent Events [W3C-20150203] transport strategy to push filtered Event Notifications from the Event stream.

The Publisher MUST support individual parameters within the POST request body for all the parameters of a subscription. The only exception is the encoding, which is embedded in the URI. An example of this is:

```
// subtree filter = /foo
// periodic updates, every 5 seconds
POST /restconf/operations/ietf-event-notifications:
  establish-subscription HTTP/1.1
  Host: example.com
  Content-Type: application/yang-data+json

  {
    "ietf-event-notifications:input" : {
      ?stream?: ?push-data"
      ?period" : 5,
      "xpath-filter" : ?/ex:foo[starts-with(?bar?.?some']"
    }
  }
```

Should the publisher not support the requested subscription, it may reply:

```
HTTP/1.1 501 Not Implemented
Date: Mon, 23 Apr 2012 17:11:00 GMT
Server: example-server
Content-Type: application/yang.errors+xml
  <errors xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf">
    <error>
      <error-type>application</error-type>
      <error-tag>operation-not-supported</error-tag>
      <error-severity>error</error-severity>
      <error-message>Xpath filters not supported</error-message>
      <error-info>
        <supported-subscription xmlns="urn:ietf:params:xml:ns:
          netconf:datastore-push:1.0">
          <subtree-filter/>
        </supported-subscription>
      </error-info>
    </error>
  </errors>
```

with an equivalent JSON encoding representation of:

```
HTTP/1.1 501 Not Implemented
Date: Mon, 23 Apr 2012 17:11:00 GMT
Server: example-server
Content-Type: application/yang.errors+json
{
  "ietf-restconf:errors": {
    "error": {
      "error-type": "protocol",
      "error-tag": "operation-not-supported",
      "error-message": "Xpath filters not supported."
      "error-info": {
        "datastore-push:supported-subscription": {
          "subtree-filter": [null]
        }
      }
    }
  }
}
```

The following is an example of a pushed Event Notification data for the Subscription above. It contains a subtree with root foo that contains a leaf called bar:

XML encoding representation:

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf">
  <subscription-id xmlns="urn:ietf:params:xml:ns:restconf:
    datastore-push:1.0">
    my-sub
  </subscription-id>
  <eventTime>2015-03-09T19:14:56.233Z</eventTime>
  <datastore-contents xmlns="urn:ietf:params:xml:ns:restconf:
    datastore-push:1.0">
    <foo xmlns="http://example.com/yang-push/1.0">
      <bar>some_string</bar>
    </foo>
  </datastore-contents>
</notification>
```

Or with the equivalent YANG over JSON encoding representation as defined in [RFC7951]:

```
{
  "ietf-restconf:notification": {
    "datastore-push:subscription-id": "my-sub",
    "eventTime": "2015-03-09T19:14:56.233Z",
    "datastore-push:datastore-contents": {
      "example-mod:foo": { "bar": "some_string" }
    }
  }
}
```

To modify a Subscription, the subscriber issues another POST request on the provided URI using the same subscription-id as in the original request. For example, to modify the update period to 10 seconds, the subscriber may send:

```
POST /restconf/operations/ietf-event-notifications:
modify-subscription HTTP/1.1
Host: example.com
Content-Type: application/yang-data+json

{
  "ietf-event-notifications:input" : {
    ?subscription-id?: 100,
    ?period" : 10
  }
}
```

To delete a Subscription, the Subscriber issues a DELETE request on the provided URI using the same subscription-id as in the original request

#### 4.2. Event Notification over HTTP2

The basic encoding will look as below. It will consists of a JSON representation wrapped in an HTTP2 header.

HyperText Transfer Protocol 2

Stream: HEADERS, Stream ID: 5

Header: :method: POST

Stream: HEADERS, Stream ID: 5

```
{
  "ietf-yangpush:notification": {
    "datastore-push:subscription-id": "my-sub",
    "eventTime": "2015-03-09T19:14:56.233Z",
    "datastore-push:datastore-contents": {
      "foo": { "bar": "some_string" }
    }
  }
}
```

#### 5. Security Considerations

Subscriptions could be used to intentionally or accidentally overload the resources of a Publisher. For this reason, it is important that the Publisher has the ability to prioritize the establishment and push of Event Notifications where there might be resource exhaust potential. In addition, a server needs to be able to suspend existing Subscriptions when needed. When this occurs, the subscription status must be updated accordingly and the Receivers notified.

A Subscription could be used to attempt retrieve information for which a Receiver has no authorized access. Therefore it is important that data pushed via a Subscription is authorized equivalently with regular data retrieval operations. Data being pushed to a Receiver needs therefore to be filtered accordingly, just like if the data were being retrieved on-demand. The Netconf Authorization Control Model [RFC6536] applies even though the transport is not NETCONF.

One or more Publishers of Configured Subscriptions could be used to overwhelm a Receiver which doesn't even support Subscriptions. There are two protections here. First Event Notifications for Configured Subscriptions MUST only be transmittable over Encrypted transports. Clients which do not want pushed Event Notifications need only

terminate or refuse any transport sessions from the Publisher. Second, the HTTP transport augmentation on the Receiver must send an HTTP 200 OK to a subscription started notification before the Publisher starts streaming any events.

One or more Publishers could overwhelm a Receiver which is unable to control or handle the volume of Event Notifications received. In deployments where this might be a concern, HTTP2 transport such as HTTP2) should be selected.

## 6. Acknowledgments

We wish to acknowledge the helpful contributions, comments, and suggestions that were received from: Susan Hares, Tim Jenkins, Balazs Lengyel, Kent Watsen, Michael Scharf, and Guangying Zheng.

## 7. References

### 7.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [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, <<http://www.rfc-editor.org/info/rfc6520>>.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <<http://www.rfc-editor.org/info/rfc6536>>.
- [RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <<http://www.rfc-editor.org/info/rfc7540>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<http://www.rfc-editor.org/info/rfc8040>>.



[sn]            Voit, E., Clemm, A., Gonzalez Prieto, A., Prasad Tripathy, A., and E. Nilsen-Nygaard, "Subscribing to Event Notifications", February 2017, <<https://datatracker.ietf.org/doc/draft-ietf-netconf-subscribed-notifications/>>.

## 7.2. Informative References

- [RFC7923]    Voit, E., Clemm, A., and A. Gonzalez Prieto, "Requirements for Subscription to YANG Datastores", RFC 7923, DOI 10.17487/RFC7923, June 2016, <<http://www.rfc-editor.org/info/rfc7923>>.
- [RFC7951]    Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <<http://www.rfc-editor.org/info/rfc7951>>.
- [RFC8071]    Watsen, K., "NETCONF Call Home and RESTCONF Call Home", RFC 8071, DOI 10.17487/RFC8071, February 2017, <<http://www.rfc-editor.org/info/rfc8071>>.
- [W3C-20150203]    "Server-Sent Events, World Wide Web Consortium CR CR-eventsource-20121211", February 2015, <<https://www.w3.org/TR/2015/REC-eventsource-20150203/>>.
- [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/>>.

## Appendix A. End-to-End Deployment Guidance

Several technologies are expected to be seen within a deployment to achieve security and ease-of-use requirements. These are not necessary for an implementation of this specification, but will be useful to consider when considering the operational context.

### A.1. Call Home

Pub/Sub implementations should have the ability to transparently incorporate 'call home' [RFC8071] so that secure TLS connections can originate from the desired device.

## A.2. TLS Heartbeat

HTTP sessions might not quickly allow a Subscriber to recognize when the communication path has been lost from the Publisher. To recognize this, it is possible for a Receiver to establish a TLS heartbeat [RFC6520]. In the case where a TLS heartbeat is included, it should be sent just from Receiver to Publisher. Loss of the heartbeat should result in any Subscription related TCP sessions between those endpoints being torn down. The subscription can then attempt to re-establish.

## Appendix B. Issues being worked and resolved

(To be removed by RFC editor prior to publication)

### B.1. Unresolved Issues

GRPC compatibility 1: Mechanisms for HTTP2 to GRPC mapping need to be considered. There is a good start there as this draft only uses POST, not GET. As GET is used in RESTCONF for capabilities discovery, we have some backwards compatibility issues with existing IETF drafts. Possible options to address are (1) provide a POST method for anything done by GET in RESTCONF, (2) await support of GET by GRPC, or (3) tunnel RESTCONF's GET messages within a GRPC POST.

GRPC compatibility 2: We need to expose a method against which POST is done as events begin on a stream. See Stream 7 in figure 2. Can only send traffic to a method, not a URI. URI points to a method, not a resource.

Need to add into document examples of Event streams. Document only includes yang-push examples at this point.

We need to reference the viable encodings of notifications.

## Appendix C. Changes between revisions

(To be removed by RFC editor prior to publication)

v01 - v02

- o Removed sections now redundant with [sn] and [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](mailto:evoit@cisco.com)

Alberto Gonzalez Prieto  
Cisco Systems

Email: [albertgo@cisco.com](mailto:albertgo@cisco.com)

Ambika Prasad Tripathy  
Cisco Systems

Email: [ambtripa@cisco.com](mailto:ambtripa@cisco.com)

Einar Nilsen-Nygaard  
Cisco Systems

Email: [einarnn@cisco.com](mailto:einarnn@cisco.com)

Alexander Clemm  
Huawei

Email: [ludwig@clemm.org](mailto:ludwig@clemm.org)

Internet-Draft

Restconf-Notif

March 2017

Andy Bierman  
YumaWorks

Email: [andy@yumaworks.com](mailto:andy@yumaworks.com)

NETCONF  
Internet-Draft  
Intended status: Standards Track  
Expires: December 13, 2019

E. Voit  
R. Rahman  
E. Nilsen-Nygaard  
Cisco Systems  
A. Clemm  
Huawei  
A. Bierman  
YumaWorks  
June 11, 2019

Dynamic subscription to YANG Events and Datastores over RESTCONF  
draft-ietf-netconf-restconf-notif-15

## Abstract

This document provides a RESTCONF binding to the dynamic subscription capability of both subscribed notifications and YANG-Push.

## Status of This Memo

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

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

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

This Internet-Draft will expire on December 13, 2019.

## Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Terminology . . . . .	3
3. 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 . . . . .	7
4. QoS Treatment . . . . .	9
5. Notification Messages . . . . .	10
6. YANG Tree . . . . .	10
7. YANG module . . . . .	10
8. IANA Considerations . . . . .	12
9. Security Considerations . . . . .	12
10. Acknowledgments . . . . .	13
11. References . . . . .	13
11.1. Normative References . . . . .	13
11.2. Informative References . . . . .	15
Appendix A. Examples . . . . .	16
A.1. Dynamic Subscriptions . . . . .	16
A.1.1. Establishing Dynamic Subscriptions . . . . .	16
A.1.2. Modifying Dynamic Subscriptions . . . . .	19
A.1.3. Deleting Dynamic Subscriptions . . . . .	21
A.2. Subscription State Notifications . . . . .	21
A.2.1. subscription-modified . . . . .	22
A.2.2. subscription-completed, subscription-resumed, and replay-complete . . . . .	22
A.2.3. subscription-terminated and subscription-suspended . . . . .	22
A.3. Filter Example . . . . .	23
Appendix B. Changes between revisions . . . . .	24
Authors' Addresses . . . . .	27

## 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]. Requirements for these mechanisms are captured in [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", "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 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 [RFC8446] 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 subscriber 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 3.4.

### 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. In the case when the RESTCONF binding specified by this document is used to convey the "streams" container from ietf-restconf-monitoring.yang (i.e., that feature is supported), any event streams contained therein are also expected to be present in the "streams" container of ietf-restconf-monitoring.yang.

Subscribers can learn what datastores a RESTCONF server supports by following Section 2 of [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 the 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 an appropriate error code, as shown below, transported in the HTTP response.

When an HTTP error 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 with the value being a string that corresponds to an identity associated with the error. This "error-tag" will come from one of two places. Either it will correspond to the error identities within [I-D.draft-ietf-netconf-subscribed-notifications] section 2.4.6 for general subscription errors:

error identity	uses error-tag	HTTP Code
-----	-----	-----
dscp-unavailable	invalid-value	400
encoding-unsupported	invalid-value	400
filter-unsupported	invalid-value	400
insufficient-resources	resource-denied	409
no-such-subscription	invalid-value	404
replay-unsupported	operation-not-supported	501

Or this "error-tag" will correspond to the error identities within [I-D.ietf-netconf-yang-push] Appendix A.1 for subscription errors specific to YANG datastores:

error identity	uses error-tag	HTTP Code
-----	-----	-----
cant-exclude	operation-not-supported	501
datastore-not-subscribable	invalid-value	400
no-such-subscription-resync	invalid-value	404
on-change-unsupported	operation-not-supported	501
on-change-sync-unsupported	operation-not-supported	501
period-unsupported	invalid-value	400
update-too-big	too-big	400
sync-too-big	too-big	400
unchanging-selection	operation-failed	500

- 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	delete-subscription-error
resync-subscription	resync-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 a 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 "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 "resync-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

The call flow for Server-Sent Events (SSE) is defined in Figure 1. The logical connections denoted by (a) and (b) can be a TCP connection or an HTTP2 stream (if HTTP2 is used, 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 signals the publisher to initiate the flow of notification messages which are sent in SSE [W3C-20150203] as a response to the GET. There cannot be two or more simultaneous GET requests on a subscription URI: any GET request received while there is a current GET request on the same URI MUST be rejected with HTTP error code 409.

As described in [RFC8040] Section 6.4, RESTCONF servers SHOULD NOT send the "event" or "id" fields in the SSE event notifications.

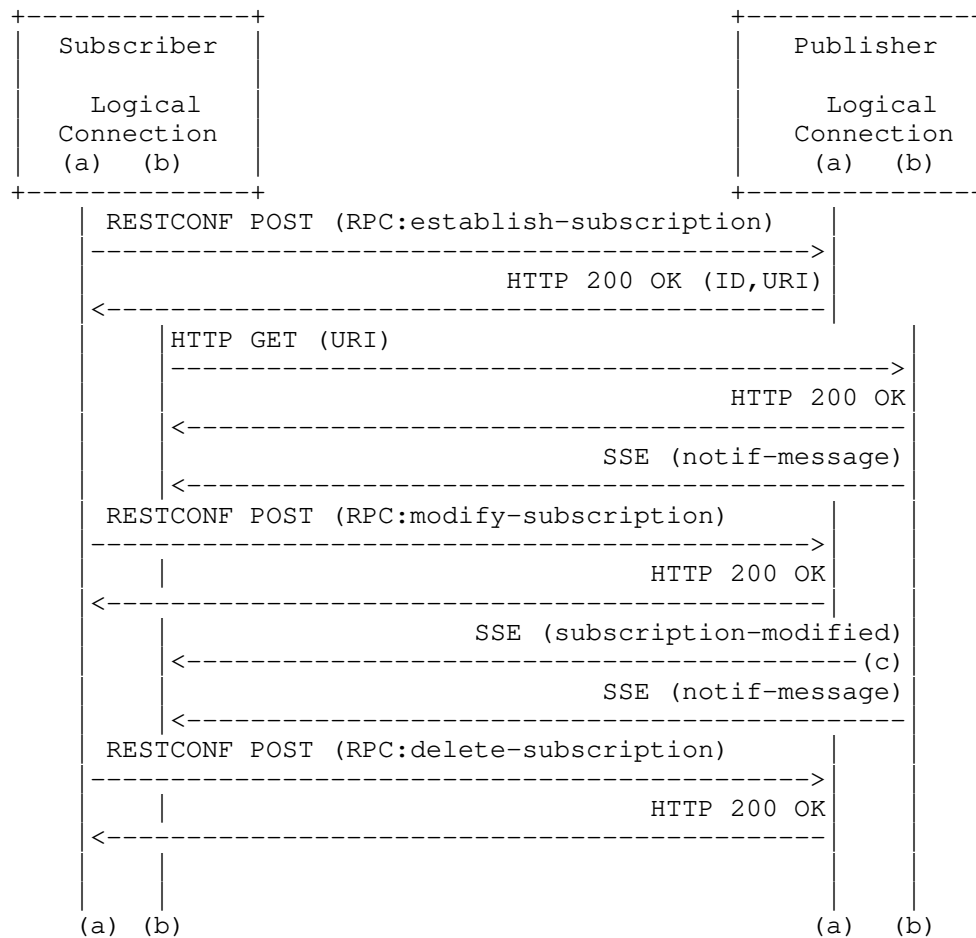


Figure 1: Dynamic with server-sent events

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, within the stream of events, the new terms of

the subscription have been applied to the notification messages. See arrow (c).

- o In addition to any required access permissions (e.g., NACM), RPCs modify-subscription, resync-subscription and delete-subscription SHOULD only be allowed by the same RESTCONF username [RFC8040] which invoked establish-subscription. Such a restriction generally serves to preserve users' privacy, but exceptions might be made for administrators that may need to modify or delete other users' subscriptions.
- o The kill-subscription RPC can be invoked by any RESTCONF username with the required administrative permissions.

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

Qos treatment for event streams is described in [I-D.draft-ietf-netconf-subscribed-notifications] Section 2.3. In addition, if HTTP2 is used, the publisher MUST:

- o take the "weighting" leaf node in [I-D.draft-ietf-netconf-subscribed-notifications], and copy it into the HTTP2 stream weight, [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.
- o set the exclusive flag, [RFC7540] section 5.3.1, to 0.

For dynamic subscriptions with the same DSCP value to a specific publisher, it is recommended that the subscriber sends all URI GET requests on a common HTTP2 session (if HTTP2 is used). Conversely, a subscriber can not use a common HTTP2 session for subscriptions with different DSCP values.

## 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 three places of [I-D.draft-ietf-netconf-subscribed-notifications].

```
module: ietf-restconf-subscribed-notifications
  augment /sn:establish-subscription/sn:output:
    +--ro uri?    inet:uri
  augment /sn:subscriptions/sn:subscription:
    +--ro uri?    inet:uri
  augment /sn:subscription-modified:
    +--ro uri?    inet:uri
```

## 7. YANG module

This module references  
[I-D.draft-ietf-netconf-subscribed-notifications].

```
<CODE BEGINS> file
  "ietf-restconf-subscribed-notifications@2019-01-11.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) 2019 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

revision 2019-01-11 {

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 as 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 [RFC8446].

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, i.e., the same RESTCONF [RFC8040] user credentials which invoked the corresponding `"establish-subscription"`, has the ability to access this resource.

The subscription URI is implementation specific and is encrypted via the use of TLS. Therefore, even if an attacker succeeds in guessing the subscription URI, a RESTCONF username [RFC8040] with the required administrative permissions must be used to be able to access or modify that subscription. It is recommended that the subscription URI values not be easily predictable.

The access permission considerations for the RPCs `modify-subscription`, `resync-subscription`, `delete-subscription` and `kill-subscription` are described in Section 3.4.

If a buggy or compromised RESTCONF subscriber sends a number of `"establish-subscription"` requests, then these subscriptions accumulate and may use up system resources. In such a situation, the publisher MAY also suspend or terminate a subset of the active subscriptions from that RESTCONF subscriber in order to reclaim resources and preserve normal operation for the other subscriptions.

## 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, Qin Wu and Robert Wilton.

## 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-21  
(work in progress), January 2019.

- [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", draft-ietf-netconf-yang-push-20 (work in progress), October 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>>.
- [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>>.
- [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>>.
- [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>>.

- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <<https://www.rfc-editor.org/info/rfc8342>>.
- [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>>.
- [W3C-20150203] Hickson, I., "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>>.
- [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>>.

- [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.

The subscription URI values used in the examples in this section are purely illustrative, and are not indicative of the expected usage which is described in Section 9.

The DSCP values are only for example purposes and are all indicated in decimal since the encoding is JSON [RFC7951].

### 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.

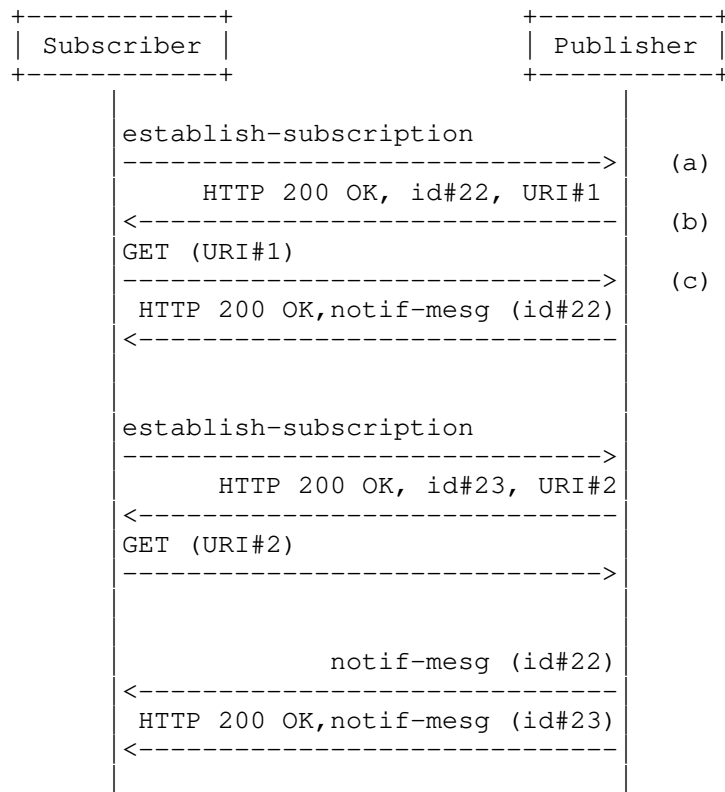


Figure 2: Multiple subscriptions over RESTCONF/HTTP

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-xpath-filter": "/example-module:foo/",
    "stream": "NETCONF",
    "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 - 400

```
{ "ietf-restconf:errors" : {
  "error" : [
    {
      "error-type": "application",
      "error-tag": "invalid-value",
      "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.1.1.2. 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.

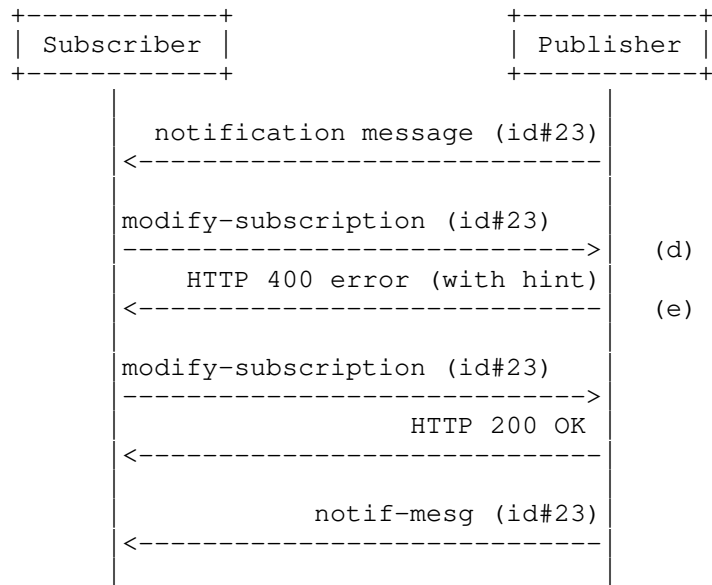


Figure 7: Interaction model for successful subscription modification

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 - 400

```
{ "ietf-restconf:errors" : {
  "error" : [
    "error-type": "application",
    "error-tag": "invalid-value",
    "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 - 404

{
  "ietf-restconf:errors" : {
    "error" : [
      "error-type": "application",
      "error-tag": "invalid-value",
      "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-error-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)

v14 - v15

- o Addressed review comments from Kent.

v13 - v14

- o Addressed review comments from IESG.

v12 - v13

- o Enhanced "error-tag" values based on SN review.

v11 - v12

- o Added text in 3.2 for expected behavior when ietf-restconf-monitoring.yang is also supported.
- o Added section 2 to the reference to draft-ietf-netconf-nmda-restconf.
- o Replaced kill-subscription-error by delete-subscription-error in section 3.3.
- o Clarified vertical lines (a) and (b) in Figure 1 of section 3.4
- o Section 3.4, 3rd bullet after Figure 1, replaced "must" with "MUST".
- o Modified text in section 3.4 regarding access to RPCs modify-subscription, resync-subscription, delete-subscription and kill-subscription.
- o Section 4, first bullet for HTTP2: replaced dscp and priority with weighting and weight.
- o Section 6, added YANG tree diagram and fixed description of the module.
- o Section 7, fixed indentation of module description statement.
- o Section 7, in YANG module changed year in copyright statement to 2019.
- o Section 8, added text on how server protects access to the subscription URI.
- o Fixed outdated references and removed unused references.
- o Fixed the instances of line too long.
- o Fixed example in Figure 3.

v10 - v11

- o Per Kent's request, added name attribute to artwork which need to be extracted

v09 - v10

- o Fixed typo for resync.
- o Added text wrt RPC permissions and RESTCONF username.

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](mailto: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



Internet Engineering Task Force  
Internet-Draft  
Obsoletes: 6536 (if approved)  
Intended status: Standards Track  
Expires: June 14, 2018

A. Bierman  
YumaWorks  
M. Bjorklund  
Tail-f Systems  
December 11, 2017

Network Configuration Access Control Module  
draft-ietf-netconf-rfc6536bis-09

Abstract

The standardization of network configuration interfaces for use with the Network Configuration Protocol (NETCONF) or RESTCONF protocol requires a structured and secure operating environment that promotes human usability and multi-vendor interoperability. There is a need for standard mechanisms to restrict NETCONF or RESTCONF protocol access for particular users to a pre-configured subset of all available NETCONF or RESTCONF protocol operations and content. This document defines such an access control model.

This document obsoletes RFC 6536.

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 <http://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 June 14, 2018.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
1.1. Terminology . . . . .	4
1.2. Changes Since RFC 6536 . . . . .	6
2. Access Control Design Objectives . . . . .	6
2.1. Access Control Points . . . . .	6
2.2. Simplicity . . . . .	7
2.3. Procedural Interface . . . . .	8
2.4. Datastore Access . . . . .	8
2.5. Users and Groups . . . . .	8
2.6. Maintenance . . . . .	8
2.7. Configuration Capabilities . . . . .	8
2.8. Identifying Security-Sensitive Content . . . . .	9
3. NETCONF Access Control Model (NACM) . . . . .	9
3.1. Introduction . . . . .	9
3.1.1. Features . . . . .	10
3.1.2. External Dependencies . . . . .	10
3.1.3. Message Processing Model . . . . .	11
3.2. Datastore Access . . . . .	14
3.2.1. Mapping New Datastores to NACM . . . . .	14
3.2.2. Access Rights . . . . .	14
3.2.3. RESTCONF Methods . . . . .	15
3.2.4. <get> and <get-config> Operations . . . . .	16
3.2.5. <edit-config> Operation . . . . .	16
3.2.6. <copy-config> Operation . . . . .	18
3.2.7. <delete-config> Operation . . . . .	18
3.2.8. <commit> Operation . . . . .	18
3.2.9. <discard-changes> Operation . . . . .	19
3.2.10. <kill-session> Operation . . . . .	19
3.3. Model Components . . . . .	19
3.3.1. Users . . . . .	19
3.3.2. Groups . . . . .	19
3.3.3. Emergency Recovery Session . . . . .	20
3.3.4. Global Enforcement Controls . . . . .	20
3.3.4.1. enable-nacm Switch . . . . .	20
3.3.4.2. read-default Switch . . . . .	20
3.3.4.3. write-default Switch . . . . .	20
3.3.4.4. exec-default Switch . . . . .	21
3.3.4.5. enable-external-groups Switch . . . . .	21
3.3.5. Access Control Rules . . . . .	21

3.4.	Access Control Enforcement Procedures . . . . .	22
3.4.1.	Initial Operation . . . . .	22
3.4.2.	Session Establishment . . . . .	22
3.4.3.	"access-denied" Error Handling . . . . .	22
3.4.4.	Incoming RPC Message Validation . . . . .	23
3.4.5.	Data Node Access Validation . . . . .	25
3.4.6.	Outgoing <notification> Authorization . . . . .	27
3.5.	Data Model Definitions . . . . .	30
3.5.1.	Data Organization . . . . .	30
3.5.2.	YANG Module . . . . .	30
3.6.	IANA Considerations . . . . .	40
3.7.	Security Considerations . . . . .	41
3.7.1.	NACM Configuration and Monitoring Considerations . .	41
3.7.2.	General Configuration Issues . . . . .	43
3.7.3.	Data Model Design Considerations . . . . .	45
4.	References . . . . .	45
4.1.	Normative References . . . . .	45
4.2.	Informative References . . . . .	46
Appendix A.	Change Log . . . . .	47
A.1.	v08 to v09 . . . . .	47
A.2.	v07 to v08 . . . . .	47
A.3.	v06 to v07 . . . . .	47
A.4.	v05 to v06 . . . . .	47
A.5.	v04 to v05 . . . . .	47
A.6.	v03 to v04 . . . . .	47
A.7.	v02 to v03 . . . . .	48
A.8.	v01 to v02 . . . . .	48
A.9.	v00 to v01 . . . . .	48
A.10.	v00 . . . . .	48
Appendix B.	Usage Examples . . . . .	48
B.1.	<groups> Example . . . . .	48
B.2.	Module Rule Example . . . . .	49
B.3.	Protocol Operation Rule Example . . . . .	51
B.4.	Data Node Rule Example . . . . .	53
B.5.	Notification Rule Example . . . . .	55
Authors' Addresses	. . . . .	55

## 1. Introduction

The NETCONF and RESTCONF protocols do not provide any standard mechanisms to restrict the protocol operations and content that each user is authorized to access.

There is a need for interoperable management of the controlled access to administrator-selected portions of the available NETCONF or RESTCONF content within a particular server.

This document addresses access control mechanisms for the Operations and Content layers of NETCONF, as defined in [RFC6241], and RESTCONF, as defined in [RFC8040]. It contains three main sections:

1. Access Control Design Objectives
2. NETCONF Access Control Model (NACM)
3. YANG Data Model (ietf-netconf-acm.yang)

YANG version 1.1 [RFC7950] adds two new constructs that need special access control handling. The "action" statement is similar to the "rpc" statement, except it is located within a data node. The "notification" statement can also be located within a data node.

### 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.

The following terms are defined in [I-D.ietf-netmod-revised-datastores] and are not redefined here:

- o datastore
- o configuration datastore
- o conventional configuration datastore
- o candidate configuration datastore
- o running configuration datastore
- o startup configuration datastore
- o operational state datastore
- o client
- o server

The following terms are defined in [RFC6241] and are not redefined here:

- o protocol operation

- o session
- o user

The following terms are defined in [RFC7950] and are not redefined here:

- o action
- o data node
- o data definition statement

The following terms are defined in [RFC8040] and are not redefined here:

- o data resource
- o data store resource
- o operation resource
- o target resource

The following term is defined in [RFC7230] and is not redefined here:

- o request URI

The following terms are used throughout this document:

access control: A security feature provided by the server that allows an administrator to restrict access to a subset of all protocol operations and data, based on various criteria.

access control model (ACM): A conceptual model used to configure and monitor the access control procedures desired by the administrator to enforce a particular access control policy.

access control rule: The criterion used to determine if a particular access operation will be permitted or denied.

access operation: How a request attempts to access a conceptual object. One of "none", "read", "create", "delete", "update", or "execute".

data node hierarchy: The hierarchy of data nodes that identifies the specific "action" or "notification" node in the data store.

recovery session: A special administrative session that is given unlimited NETCONF access and is exempt from all access control enforcement. The mechanism(s) used by a server to control and identify whether or not a session is a recovery session are implementation specific and outside the scope of this document.

write access: A shorthand for the "create", "delete", and "update" access operations.

## 1.2. Changes Since RFC 6536

The NACM procedures and data model have been updated to support new data modeling capabilities in the version 1.1. of the YANG data modeling language. The "action" and "notification" statements can be used within data nodes to define data-model specific operations and notifications.

An important use-case for these new YANG statements is the increased access control granularity that can be achieved over top-level "rpc" and "notification" statements. The new "action" and "notification" statements are used within data nodes, and access to the action or notification can be restricted to specific instances of these data nodes.

Support for the RESTCONF protocol has been added. The RESTCONF operations are similar to the NETCONF operations, so a simple mapping to the existing NACM procedures and data model is possible.

The data node access behavior for path matches has been clarified to also include matching descendant nodes of the specified path.

The <edit-config> operation access rights behavior has been clarified to indicate that write access is not required for data nodes that are implicitly modified through side-effects (such as the evaluation of YANG when-stmts, or data nodes implicitly deleted when creating a data node under a different branch under a YANG choice-stmt).

## 2. Access Control Design Objectives

This section documents the design objectives for the NETCONF Access Control Model presented in Section 3.

### 2.1. Access Control Points

NETCONF allows server implementors to add new custom protocol operations, and the YANG Data Modeling Language supports this feature. These operations can be defined in standard or proprietary YANG modules.

It is not possible to design an ACM for NETCONF that only focuses on a static set of standard protocol operations defined by the NETCONF protocol itself, like some other protocols. Since few assumptions can be made about an arbitrary protocol operation, the NETCONF architectural server components need to be protected at three conceptual control points.

These access control points, described in Figure 1, are as follows:

protocol operation: Permission to invoke specific protocol operations.

datastore: Permission to read and/or alter specific data nodes within any datastore.

notification: Permission to receive specific notification event types.

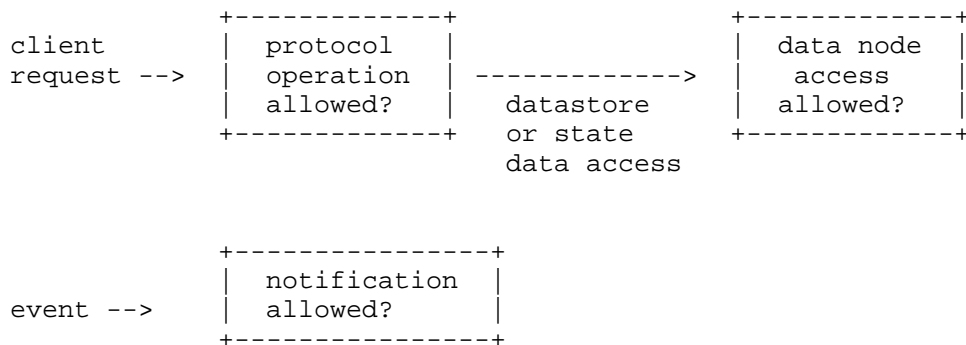


Figure 1

## 2.2. Simplicity

There is concern that a complicated ACM will not be widely deployed because it is too hard to use. Configuration of the access control system needs to be as simple as possible. Simple and common tasks need to be easy to configure and require little expertise or domain-specific knowledge. Complex tasks are possible using additional mechanisms, which may require additional expertise.

A single set of access control rules ought to be able to control all types of NETCONF protocol operation invocation, all datastore access, and all notification events.

Access control ought to be defined with a small and familiar set of permissions, while still allowing full control of datastore access.

### 2.3. Procedural Interface

The NETCONF protocol uses a remote procedure call model and an extensible set of protocol operations. Access control for any possible protocol operation is necessary.

### 2.4. Datastore Access

It is necessary to control access to specific nodes and subtrees within the datastore, regardless of which protocol operation, standard or proprietary, was used to access the datastore.

### 2.5. Users and Groups

It is necessary that access control rules for a single user or a configurable group of users can be configured.

The ACM needs to support the concept of administrative groups, to support the well-established distinction between a root account and other types of less-privileged conceptual user accounts. These groups need to be configurable by the administrator.

It is necessary that the user-to-group mapping can be delegated to a central server, such as a RADIUS server [RFC2865][RFC5607]. Since authentication is performed by the transport layer and RADIUS performs authentication and service authorization at the same time, the underlying transport protocol needs to be able to report a set of group names associated with the user to the server. It is necessary that the administrator can disable the usage of these group names within the ACM.

### 2.6. Maintenance

It ought to be possible to disable part or all of the access control model enforcement procedures without deleting any access control rules.

### 2.7. Configuration Capabilities

Suitable configuration and monitoring mechanisms are needed to allow an administrator to easily manage all aspects of the ACM's behavior. A standard data model, suitable for use with the <edit-config> protocol operation, needs to be available for this purpose.

Access control rules to restrict access operations on specific subtrees within the configuration datastore need to be supported.



## 2.8. Identifying Security-Sensitive Content

One of the most important aspects of the data model documentation, and biggest concerns during deployment, is the identification of security-sensitive content. This applies to protocol operations in NETCONF, not just data and notifications.

It is mandatory for security-sensitive objects to be documented in the Security Considerations section of an RFC. This is nice, but it is not good enough, for the following reasons:

- o This documentation-only approach forces administrators to study the RFC and determine if there are any potential security risks introduced by a new data model.
- o If any security risks are identified, then the administrator must study some more RFC text and determine how to mitigate the security risk(s).
- o The ACM on each server must be configured to mitigate the security risks, e.g., require privileged access to read or write the specific data identified in the Security Considerations section.
- o If the ACM is not pre-configured, then there will be a time window of vulnerability after the new data model is loaded and before the new access control rules for that data model are configured, enabled, and debugged.

Often, the administrator just wants to disable default access to the secure content, so no inadvertent or malicious changes can be made to the server. This allows the default rules to be more lenient, without significantly increasing the security risk.

A data model designer needs to be able to use machine-readable statements to identify content that needs to be protected by default. This will allow client and server tools to automatically identify data-model-specific security risks, by denying access to sensitive data unless the user is explicitly authorized to perform the requested access operation.

## 3. NETCONF Access Control Model (NACM)

### 3.1. Introduction

This section provides a high-level overview of the access control model structure. It describes the NETCONF protocol message processing model and the conceptual access control requirements within that model.

### 3.1.1. Features

The NACM data model provides the following features:

- o Independent control of remote procedure call (RPC), action, data, and notification access.
- o Simple access control rules configuration data model that is easy to use.
- o The concept of an emergency recovery session is supported, but configuration of the server for this purpose is beyond the scope of this document. An emergency recovery session will bypass all access control enforcement, in order to allow it to initialize or repair the NACM configuration.
- o A simple and familiar set of datastore permissions is used.
- o Support for YANG security tagging (e.g., "nacm:default-deny-write" statement) allows default security modes to automatically exclude sensitive data.
- o Separate default access modes for read, write, and execute permissions.
- o Access control rules are applied to configurable groups of users.
- o The access control enforcement procedures can be disabled during operation, without deleting any access control rules, in order to debug operational problems.
- o Access control rules are simple to configure.
- o The number of denied protocol operation requests and denied datastore write requests can be monitored by the client.
- o Simple unconstrained YANG instance identifiers are used to configure access control rules for specific data nodes.

### 3.1.2. External Dependencies

The NETCONF protocol [RFC6241] and RESTCONF protocol [RFC8040] are used for network management purposes within this document.

The YANG Data Modeling Language [RFC7950] is used to define the data models for use with the NETCONF or RESTCONF protocols. YANG is also used to define the data model in this document.

### 3.1.3. Message Processing Model

The following diagram shows the conceptual message flow model, including the points at which access control is applied during NETCONF message processing.

RESTCONF operations are mapped to the access control model based on the HTTP method and resource class used in the operation. For example, a POST method on a data resource is considered "write data node" access, but a POST method on an operation resource is considered "operation" access.

The new "pre-read data acc. ctl" boxes in the diagram below refer to group read access as it relates to data node ancestors of an action or notification. As an example, if an action is defined as /interfaces/interface/reset-interface, the group must be authorized to read /interfaces and /interfaces/interface, and execute on /interfaces/interface/reset-interface.

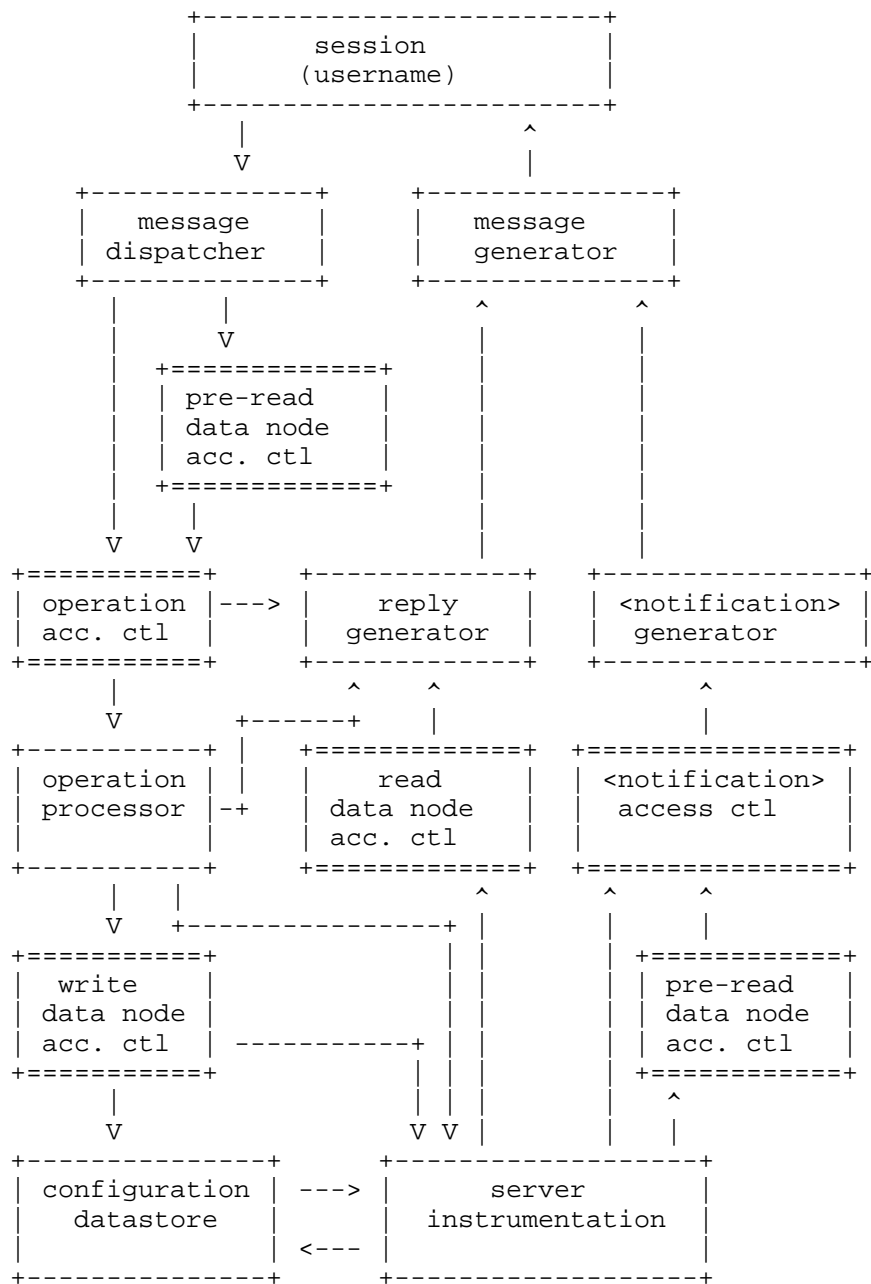


Figure 2

The following high-level sequence of conceptual processing steps is executed for each received <rpc> message, if access control enforcement is enabled:

- o For each active session, access control is applied individually to all <rpc> messages (except <close-session>) received by the server, unless the session is identified as a recovery session.
- o If the <action> operation defined in [RFC7950] is invoked, then read access is required for all instances in the hierarchy of data nodes that identifies the specific action in the datastore, and execute access is required for the action node. If the user is not authorized to read all the specified data nodes and execute the action, then the request is rejected with an "access-denied" error.
- o Otherwise, if the user is not authorized to execute the specified protocol operation, then the request is rejected with an "access-denied" error.
- o If a datastore is accessed by the protocol operation, then the server checks if the client is authorized to access the nodes in the datastore. If the user is not authorized to perform the requested access operation on the requested data, then the request is rejected with an "access-denied" error.

The following sequence of conceptual processing steps is executed for each generated notification event, if access control enforcement is enabled:

- o Server instrumentation generates a notification for a particular subscription.
- o If the notification statement is specified within a data subtree, as specified in [RFC7950], then read access is required for all instances in the hierarchy of data nodes that identifies the specific notification in the datastore, and read access is required for the notification node. If the user is not authorized to read all the specified data nodes and the notification node, then the notification is dropped for that subscription.
- o If the notification statement is a top-level statement, the notification access control enforcer checks the notification event type, and if it is one that the user is not authorized to read, then the notification is dropped for that subscription.

### 3.2. Datastore Access

The same access control rules apply to all datastores that support NACM, for example, the candidate configuration datastore or the running configuration datastore.

All conventional configuration datastores and the operational state datastore are controlled by NACM. Local or remote files or datastores accessed via the <url> parameter are not controlled by NACM.

#### 3.2.1. Mapping New Datastores to NACM

It is possible that new datastores will be defined over time for use with the NETCONF protocol. NACM MAY be applied to other datastores that have similar access rights as defined in NACM. To apply NACM to a new datastore, the new datastore specification needs to define how it maps to the NACM CRUDX access rights. It is possible only a subset of the NACM access rights would be applicable. For example, only retrieval access control would be needed for a read-only datastore. Operations and access rights not supported by the NACM CRUDX model are outside the scope of this document. A datastore does not need to use NACM, e.g., the datastore specification defines something else, or does not use access control.

#### 3.2.2. Access Rights

A small set of hard-wired datastore access rights is needed to control access to all possible protocol operations, including vendor extensions to the standard protocol operation set.

The "CRUDX" model can support all protocol operations:

- o Create: allows the client to add a new data node instance to a datastore.
- o Read: allows the client to read a data node instance from a datastore or receive the notification event type.
- o Update: allows the client to update an existing data node instance in a datastore.
- o Delete: allows the client to delete a data node instance from a datastore.
- o eXec: allows the client to execute the operation.

### 3.2.3. RESTCONF Methods

The RESTCONF protocol utilizes HTTP methods to perform datastore operations, similar to the NETCONF protocol. The NACM procedures were originally written for NETCONF protocol operations so the RESTCONF methods are mapped to NETCONF operations for the purpose of access control processing. The enforcement procedures described within this document apply to both protocols unless explicitly stated otherwise.

The request URI needs to be considered when processing RESTCONF requests on data resources:

- o For HEAD and GET requests, any data nodes which are ancestor nodes of the target resource are considered to be part of the retrieval request for access control purposes.
- o For PUT, PATCH, and DELETE requests, any data nodes which are ancestor nodes of the target resource are not considered to be part of the edit request for access control purposes. The access operation for these nodes is considered to be "none". The edit begins at the target resource.
- o For POST requests on data resources, any data nodes which are specified in the request URI, including the target resource, are not considered to be part of the edit request for access control purposes. The access operation for these nodes is considered to be "none". The edit begins at a child node of the target resource, specified in the message body.

Not all RESTCONF methods are subject to access control. The following table specifies how each method is mapped to NETCONF protocol operations. The value "none" indicates that NACM is not applied at all to the specific RESTCONF method.

method	resource class	NETCONF operation	Access operation
OPTIONS	all	none	none
HEAD	all	<get>, <get-config>	read
GET	all	<get>, <get-config>	read
POST	datastore, data	<edit-config>	create
POST	operation	specified operation	execute
PUT	data	<edit-config>	create, update
PUT	datastore	<copy-config>	update
PATCH	data, datastore	<edit-config>	update
DELETE	data	<edit-config>	delete

Table 1: Mapping RESTCONF Methods to NETCONF

#### 3.2.4. <get> and <get-config> Operations

The NACM access rights are not directly coupled to the <get> and <get-config> protocol operations, but apply to all <rpc> operations that would result in a "read" access operation to the target datastore. This section describes how these access rights apply to the specific access operations supported by the <get> and <get-config> protocol operations.

Data nodes to which the client does not have read access are silently omitted, along with any descendants, from the <rpc-reply> message. This is done to allow NETCONF filters for <get> and <get-config> to function properly, instead of causing an "access-denied" error because the filter criteria would otherwise include unauthorized read access to some data nodes. For NETCONF filtering purposes, the selection criteria is applied to the subset of nodes that the user is authorized to read, not the entire datastore.

#### 3.2.5. <edit-config> Operation

The NACM access rights are not directly coupled to the <edit-config> "operation" attribute, although they are similar. Instead, a NACM access right applies to all protocol operations that would result in a particular access operation to the target datastore. This section describes how these access rights apply to the specific access operations supported by the <edit-config> protocol operation.

If the effective access operation is "none" (i.e., default-operation="none") for a particular data node, then no access control is applied to that data node. This is required to allow access to a subtree within a larger data structure. For example, a user may be



authorized to create a new `"/interfaces/interface"` list entry but not be authorized to create or delete its parent container (`"/interfaces"`). If the `"/interfaces"` container already exists in the target datastore, then the effective operation will be "none" for the `"/interfaces"` node if an `"/interfaces/interface"` list entry is edited.

If the protocol operation would result in the creation of a datastore node and the user does not have "create" access permission for that node, the protocol operation is rejected with an "access-denied" error.

If the protocol operation would result in the deletion of a datastore node and the user does not have "delete" access permission for that node, the protocol operation is rejected with an "access-denied" error.

If the protocol operation would result in the modification of a datastore node and the user does not have "update" access permission for that node, the protocol operation is rejected with an "access-denied" error.

A "merge" or "replace" `<edit-config>` operation may include data nodes that do not alter portions of the existing datastore. For example, a container or list node may be present for naming purposes but does not actually alter the corresponding datastore node. These unaltered data nodes are ignored by the server and do not require any access rights by the client.

A "merge" `<edit-config>` operation may include data nodes but not include particular child data nodes that are present in the datastore. These missing data nodes within the scope of a "merge" `<edit-config>` operation are ignored by the server and do not require any access rights by the client.

The contents of specific restricted datastore nodes MUST NOT be exposed in any `<rpc-error>` elements within the reply.

An `<edit-config>` operation may cause data nodes to be implicitly created or deleted as an implicit side-effect of a requested operation. For example, a YANG `when-stmt` expression may evaluate to a different result, causing data nodes to be deleted, or created with default values; or if a data node is created under one branch of a YANG `choice-stmt`, then all data nodes under the other branches are implicitly removed. No NACM access rights are required on any data nodes that are implicitly changed as a side effect of another allowed operation.

### 3.2.6. <copy-config> Operation

Access control for the <copy-config> protocol operation requires special consideration because the administrator may be replacing the entire target datastore.

If the source of the <copy-config> protocol operation is the running configuration datastore and the target is the startup configuration datastore, the client is only required to have permission to execute the <copy-config> protocol operation.

Otherwise:

- o If the source of the <copy-config> operation is a datastore, then data nodes to which the client does not have read access are silently omitted.
- o If the target of the <copy-config> operation is a datastore, the client needs access to the modified nodes, specifically:
  - \* If the protocol operation would result in the creation of a datastore node and the user does not have "create" access permission for that node, the protocol operation is rejected with an "access-denied" error.
  - \* If the protocol operation would result in the deletion of a datastore node and the user does not have "delete" access permission for that node, the protocol operation is rejected with an "access-denied" error.
  - \* If the protocol operation would result in the modification of a datastore node and the user does not have "update" access permission for that node, the protocol operation is rejected with an "access-denied" error.

### 3.2.7. <delete-config> Operation

Access to the <delete-config> protocol operation is denied by default. The "exec-default" leaf does not apply to this protocol operation. Access control rules must be explicitly configured to allow invocation by a non-recovery session.

### 3.2.8. <commit> Operation

The server MUST determine the exact nodes in the running configuration datastore that are actually different and only check "create", "update", and "delete" access permissions for this set of nodes, which could be empty.

For example, if a session can read the entire datastore but only change one leaf, that session needs to be able to edit and commit that one leaf.

#### 3.2.9. <discard-changes> Operation

The client is only required to have permission to execute the <discard-changes> protocol operation. No datastore permissions are needed.

#### 3.2.10. <kill-session> Operation

The <kill-session> operation does not directly alter a datastore. However, it allows one session to disrupt another session that is editing a datastore.

Access to the <kill-session> protocol operation is denied by default. The "exec-default" leaf does not apply to this protocol operation. Access control rules must be explicitly configured to allow invocation by a non-recovery session.

### 3.3. Model Components

This section defines the conceptual components related to the access control model.

#### 3.3.1. Users

A "user" is the conceptual entity that is associated with the access permissions granted to a particular session. A user is identified by a string that is unique within the server.

As described in [RFC6241], the username string is derived from the transport layer during session establishment. If the transport layer cannot authenticate the user, the session is terminated.

#### 3.3.2. Groups

Access to a specific NETCONF protocol operation is granted to a session, associated with a group, not a user.

A group is identified by its name. All group names are unique within the server.

Access control is applied at the level of groups. A group contains zero or more group members.

A group member is identified by a username string.

The same user can be a member of multiple groups.

### 3.3.3. Emergency Recovery Session

The server MAY support a recovery session mechanism, which will bypass all access control enforcement. This is useful for restricting initial access and repairing a broken access control configuration.

### 3.3.4. Global Enforcement Controls

There are five global controls that are used to help control how access control is enforced.

#### 3.3.4.1. enable-nacm Switch

A global "enable-nacm" on/off switch is provided to enable or disable all access control enforcement. When this global switch is set to "true", then all requests are checked against the access control rules and only permitted if configured to allow the specific access request. When this global switch is set to "false", then all access requested are permitted.

#### 3.3.4.2. read-default Switch

An on/off "read-default" switch is provided to enable or disable default access to receive data in replies and notifications. When the "enable-nacm" global switch is set to "true", then this global switch is relevant if no matching access control rule is found to explicitly permit or deny read access to the requested datastore data or notification event type.

When this global switch is set to "permit" and no matching access control rule is found for the datastore read or notification event requested, then access is permitted.

When this global switch is set to "deny" and no matching access control rule is found for the datastore read or notification event requested, then access is denied.

#### 3.3.4.3. write-default Switch

An on/off "write-default" switch is provided to enable or disable default access to alter configuration data. When the "enable-nacm" global switch is set to "true", then this global switch is relevant if no matching access control rule is found to explicitly permit or deny write access to the requested datastore data.

When this global switch is set to "permit" and no matching access control rule is found for the datastore write requested, then access is permitted.

When this global switch is set to "deny" and no matching access control rule is found for the datastore write requested, then access is denied.

#### 3.3.4.4. exec-default Switch

An on/off "exec-default" switch is provided to enable or disable default access to execute protocol operations. When the "enable-nacm" global switch is set to "true", then this global switch is relevant if no matching access control rule is found to explicitly permit or deny access to the requested NETCONF protocol operation.

When this global switch is set to "permit" and no matching access control rule is found for the NETCONF protocol operation requested, then access is permitted.

When this global switch is set to "deny" and no matching access control rule is found for the NETCONF protocol operation requested, then access is denied.

#### 3.3.4.5. enable-external-groups Switch

When this global switch is set to "true", the group names reported by the transport layer for a session are used together with the locally configured group names to determine the access control rules for the session.

When this switch is set to "false", the group names reported by the transport layer are ignored by NACM.

#### 3.3.5. Access Control Rules

There are four types of rules available in NACM:

module rule: controls access for definitions in a specific YANG module, identified by its name.

protocol operation rule: controls access for a specific protocol operation, identified by its YANG module and name.

data node rule: controls access for a specific data node and its descendants, identified by its path location within the conceptual XML document for the data node.

notification rule: controls access for a specific notification event type, identified by its YANG module and name.

### 3.4. Access Control Enforcement Procedures

There are seven separate phases that need to be addressed, four of which are related to the NETCONF message processing model (Section 3.1.3). In addition, the initial startup mode for a NETCONF server, session establishment, and "access-denied" error-handling procedures also need to be considered.

The server **MUST** use the access control rules in effect at the time it starts processing the message. The same access control rules **MUST** stay in effect for the processing of the entire message.

#### 3.4.1. Initial Operation

Upon the very first startup of the NETCONF server, the access control configuration will probably not be present. If it isn't, a server **MUST NOT** allow any write access to any session role except a recovery session.

Access rules are enforced any time a request is initiated from a user session. Access control is not enforced for server-initiated access requests, such as the initial load of the running configuration datastore, during bootup.

#### 3.4.2. Session Establishment

The access control model applies specifically to the well-formed XML content transferred between a client and a server after session establishment has been completed and after the <hello> exchange has been successfully completed.

Once session establishment is completed and a user has been authenticated, the transport layer reports the username and a possibly empty set of group names associated with the user to the NETCONF server. The NETCONF server will enforce the access control rules, based on the supplied username, group names, and the configuration data stored on the server.

#### 3.4.3. "access-denied" Error Handling

The "access-denied" error-tag is generated when the access control system denies access to either a request to invoke a protocol operation or a request to perform a particular access operation on the configuration datastore.

A server MUST NOT include any information the client is not allowed to read in any `<error-info>` elements within the `<rpc-error>` response.

#### 3.4.4. Incoming RPC Message Validation

The diagram below shows the basic conceptual structure of the access control processing model for incoming NETCONF `<rpc>` messages within a server.

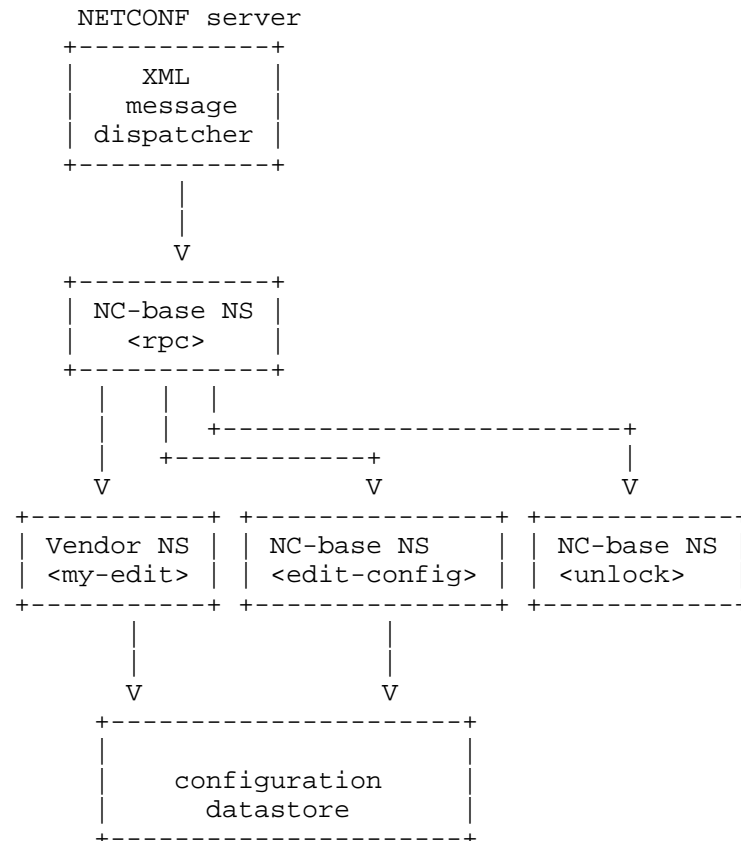


Figure 3

Access control begins with the message dispatcher.

After the server validates the `<rpc>` element and determines the namespace URI and the element name of the protocol operation being requested, the server verifies that the user is authorized to invoke the protocol operation.

The server MUST separately authorize every protocol operation by following these steps:

1. If the "enable-nacm" leaf is set to "false", then the protocol operation is permitted.
2. If the requesting session is identified as a recovery session, then the protocol operation is permitted.
3. If the requested operation is the NETCONF <close-session> protocol operation, then the protocol operation is permitted.
4. Check all the "group" entries for ones that contain a "user-name" entry that equals the username for the session making the request. If the "enable-external-groups" leaf is "true", add to these groups the set of groups provided by the transport layer.
5. If no groups are found, continue with step 10.
6. Process all rule-list entries, in the order they appear in the configuration. If a rule-list's "group" leaf-list does not match any of the user's groups, proceed to the next rule-list entry.
7. For each rule-list entry found, process all rules, in order, until a rule that matches the requested access operation is found. A rule matches if all of the following criteria are met:
  - \* The rule's "module-name" leaf is "\*" or equals the name of the YANG module where the protocol operation is defined.
  - \* The rule does not have a "rule-type" defined or the "rule-type" is "protocol-operation" and the "rpc-name" is "\*" or equals the name of the requested protocol operation.
  - \* The rule's "access-operations" leaf has the "exec" bit set or has the special value "\*".
8. If a matching rule is found, then the "action" leaf is checked. If it is equal to "permit", then the protocol operation is permitted; otherwise, it is denied.
9. At this point, no matching rule was found in any rule-list entry.
10. If the requested protocol operation is defined in a YANG module advertised in the server capabilities and the "rpc" statement



contains a "nacm:default-deny-all" statement, then the protocol operation is denied.

11. If the requested protocol operation is the NETCONF <kill-session> or <delete-config>, then the protocol operation is denied.
12. If the "exec-default" leaf is set to "permit", then permit the protocol operation; otherwise, deny the request.

If the user is not authorized to invoke the protocol operation, then an <rpc-error> is generated with the following information:

error-tag: access-denied

error-path: Identifies the requested protocol operation. The following example represents the <edit-config> protocol operation in the NETCONF base namespace:

```
<error-path
  xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  /nc:rpc/nc:edit-config
</error-path>
```

If a datastore is accessed, either directly or as a side effect of the protocol operation, then the server MUST intercept the access operation and make sure the user is authorized to perform the requested access operation on the specified data, as defined in Section 3.4.5.

### 3.4.5. Data Node Access Validation

If a data node within a datastore is accessed, or an action or notification tied to a data node, then the server MUST ensure that the user is authorized to perform the requested "read", "create", "update", "delete", or "execute" access operation on the specified data node.

If an action is requested to be executed, the server MUST ensure that the user is authorized to perform the "execute" access operation on the requested action.

If a notification tied to a data node is generated, the server MUST ensure that the user is authorized to perform the "read" access operation on the requested notification.

The data node access request is authorized by following these steps:

1. If the "enable-nacm" leaf is set to "false", then the access operation is permitted.
2. If the requesting session is identified as a recovery session, then the access operation is permitted.
3. Check all the "group" entries for ones that contain a "user-name" entry that equals the username for the session making the request. If the "enable-external-groups" leaf is "true", add to these groups the set of groups provided by the transport layer.
4. If no groups are found, continue with step 9.
5. Process all rule-list entries, in the order they appear in the configuration. If a rule-list's "group" leaf-list does not match any of the user's groups, proceed to the next rule-list entry.
6. For each rule-list entry found, process all rules, in order, until a rule that matches the requested access operation is found. A rule matches if all of the following criteria are met:
  - \* The rule's "module-name" leaf is "\*" or equals the name of the YANG module where the requested data node is defined.
  - \* The rule does not have a "rule-type" defined or the "rule-type" is "data-node" and the "path" matches the requested data node, action node, or notification node. A path is considered to match if the requested node is the node specified by the path, or is a descendant node of the path.
  - \* For a "read" access operation, the rule's "access-operations" leaf has the "read" bit set or has the special value "\*".
  - \* For a "create" access operation, the rule's "access-operations" leaf has the "create" bit set or has the special value "\*".
  - \* For a "delete" access operation, the rule's "access-operations" leaf has the "delete" bit set or has the special value "\*".
  - \* For an "update" access operation, the rule's "access-operations" leaf has the "update" bit set or has the special value "\*".

- \* For an "execute" access operation, the rule's "access-operations" leaf has the "exec" bit set or has the special value "\*".
7. If a matching rule is found, then the "action" leaf is checked. If it is equal to "permit", then the data node access is permitted; otherwise, it is denied. For a "read" access operation, "denied" means that the requested data is not returned in the reply.
  8. At this point, no matching rule was found in any rule-list entry.
  9. For a "read" access operation, if the requested data node is defined in a YANG module advertised in the server capabilities and the data definition statement contains a "nacm:default-deny-all" statement, then the requested data node and all its descendants are not included in the reply.
  10. For a "write" access operation, if the requested data node is defined in a YANG module advertised in the server capabilities and the data definition statement contains a "nacm:default-deny-write" or a "nacm:default-deny-all" statement, then the access request is denied for the data node and all its descendants.
  11. For a "read" access operation, if the "read-default" leaf is set to "permit", then include the requested data node in the reply; otherwise, do not include the requested data node in the reply.
  12. For a "write" access operation, if the "write-default" leaf is set to "permit", then permit the data node access request; otherwise, deny the request.
  13. For an "execute" access operation, if the "exec-default" leaf is set to "permit", then permit the request; otherwise, deny the request.
- 3.4.6. Outgoing <notification> Authorization

Configuration of access control rules specifically for descendant nodes of the notification event type element are outside the scope of this document. If the user is authorized to receive the notification event type, then it is also authorized to receive any data it contains.

If the notification is specified within a data subtree, as specified in [RFC7950], then read access to the notification is required. Processing continues as described in Section 3.4.5.

The following figure shows the conceptual message processing model for outgoing <notification> messages.

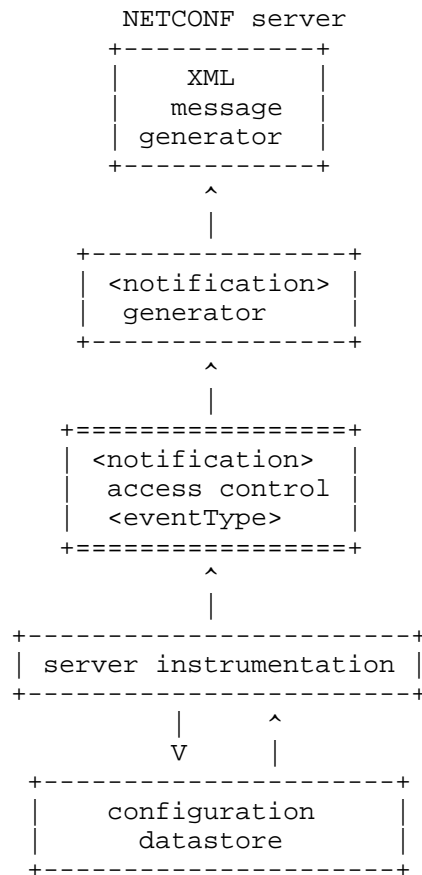


Figure 4

The generation of a notification for a specific subscription [RFC5277] is authorized by following these steps:

1. If the "enable-nacm" leaf is set to "false", then the notification is permitted.
2. If the session is identified as a recovery session, then the notification is permitted.

3. If the notification is the NETCONF <replayComplete> or <notificationComplete> event type [RFC5277], then the notification is permitted.
4. Check all the "group" entries for ones that contain a "user-name" entry that equals the username for the session making the request. If the "enable-external-groups" leaf is "true", add to these groups the set of groups provided by the transport layer.
5. If no groups are found, continue with step 10.
6. Process all rule-list entries, in the order they appear in the configuration. If a rule-list's "group" leaf-list does not match any of the user's groups, proceed to the next rule-list entry.
7. For each rule-list entry found, process all rules, in order, until a rule that matches the requested access operation is found. A rule matches if all of the following criteria are met:
  - \* The rule's "module-name" leaf is "\*" or equals the name of the YANG module where the notification is defined.
  - \* The rule does not have a "rule-type" defined or the "rule-type" is "notification" and the "notification-name" is "\*" or equals the name of the notification.
  - \* The rule's "access-operations" leaf has the "read" bit set or has the special value "\*".
8. If a matching rule is found, then the "action" leaf is checked. If it is equal to "permit", then permit the notification; otherwise, drop the notification for the associated subscription.
9. Otherwise, no matching rule was found in any rule-list entry.
10. If the requested notification is defined in a YANG module advertised in the server capabilities and the "notification" statement contains a "nacm:default-deny-all" statement, then the notification is dropped for the associated subscription.
11. If the "read-default" leaf is set to "permit", then permit the notification; otherwise, drop the notification for the associated subscription.

### 3.5. Data Model Definitions

#### 3.5.1. Data Organization

The following diagram highlights the contents and structure of the NACM YANG module.

```

module: ietf-netconf-acm
  +--rw nacm
    +--rw enable-nacm?                boolean
    +--rw read-default?               action-type
    +--rw write-default?              action-type
    +--rw exec-default?               action-type
    +--rw enable-external-groups?     boolean
    +--ro denied-operations            yang:zero-based-counter32
    +--ro denied-data-writes          yang:zero-based-counter32
    +--ro denied-notifications        yang:zero-based-counter32
    +--rw groups
      |   +--rw group* [name]
      |   |   +--rw name                group-name-type
      |   |   +--rw user-name*         user-name-type
    +--rw rule-list* [name]
      +--rw name                      string
      +--rw group*                   union
      +--rw rule* [name]
        +--rw name                    string
        +--rw module-name?            union
        +--rw (rule-type)?
          |   +--:(protocol-operation)
          |   |   +--rw rpc-name?        union
          |   +--:(notification)
          |   |   +--rw notification-name? union
          |   +--:(data-node)
          |   |   +--rw path              node-instance-identifier
        +--rw access-operations?      union
        +--rw action                  action-type
        +--rw comment?                string
  
```

#### 3.5.2. YANG Module

The following YANG module specifies the normative NETCONF content that MUST be supported by the server.

The "ietf-netconf-acm" YANG module imports typedefs from [RFC6991].

```

<CODE BEGINS> file "ietf-netconf-acm@2017-12-11.yang"
module ietf-netconf-acm {

```

```
namespace "urn:ietf:params:xml:ns:yang:ietf-netconf-acm";

prefix "nacm";

import ietf-yang-types {
  prefix yang;
}

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

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

  Author:     Andy Bierman
              <mailto:andy@yumaworks.com>

  Author:     Martin Bjorklund
              <mailto:mbj@tail-f.com>";

description
  "Network Configuration Access Control Model.

  Copyright (c) 2012, 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 "2017-12-11" {
  description
    "Added support for YANG 1.1 actions and notifications tied to
    data nodes. Clarify how NACM extensions can be used by other
    data models.";
  reference
    "RFC XXXX: Network Configuration Protocol (NETCONF)
    Access Control Model";
}
```

```
revision "2012-02-22" {
  description
    "Initial version";
  reference
    "RFC 6536: Network Configuration Protocol (NETCONF)
      Access Control Model";
}

/*
 * Extension statements
 */

extension default-deny-write {
  description
    "Used to indicate that the data model node
    represents a sensitive security system parameter.

    If present, the NETCONF server will only allow the designated
    'recovery session' to have write access to the node.  An
    explicit access control rule is required for all other users.

    If the NACM module is used, then it must be enabled (i.e.,
    /nacm/enable-nacm object equals 'true'), or this extension
    is ignored.

    The 'default-deny-write' extension MAY appear within a data
    definition statement.  It is ignored otherwise."
}

extension default-deny-all {
  description
    "Used to indicate that the data model node
    controls a very sensitive security system parameter.

    If present, the NETCONF server will only allow the designated
    'recovery session' to have read, write, or execute access to the
    node.  An explicit access control rule is required for all other
    users.

    If the NACM module is used, then it must be enabled (i.e.,
    /nacm/enable-nacm object equals 'true'), or this extension
    is ignored.

    The 'default-deny-all' extension MAY appear within a data
    definition statement, 'rpc' statement, or 'notification'
    statement.  It is ignored otherwise."
}
```



```
/*
 * Derived types
 */

typedef user-name-type {
    type string {
        length "1..max";
    }
    description
        "General Purpose Username string.";
}

typedef matchall-string-type {
    type string {
        pattern '\*';
    }
    description
        "The string containing a single asterisk '*' is used
        to conceptually represent all possible values
        for the particular leaf using this data type.";
}

typedef access-operations-type {
    type bits {
        bit create {
            description
                "Any protocol operation that creates a
                new data node.";
        }
        bit read {
            description
                "Any protocol operation or notification that
                returns the value of a data node.";
        }
        bit update {
            description
                "Any protocol operation that alters an existing
                data node.";
        }
        bit delete {
            description
                "Any protocol operation that removes a data node.";
        }
        bit exec {
            description
                "Execution access to the specified protocol operation.";
        }
    }
}
```

```
    description
      "Access Operation.";
  }

  typedef group-name-type {
    type string {
      length "1..max";
      pattern '^[^\\*].*';
    }
    description
      "Name of administrative group to which
       users can be assigned.";
  }

  typedef action-type {
    type enumeration {
      enum permit {
        description
          "Requested action is permitted.";
      }
      enum deny {
        description
          "Requested action is denied.";
      }
    }
    description
      "Action taken by the server when a particular
       rule matches.";
  }

  typedef node-instance-identifier {
    type yang:xpath1.0;
    description
      "Path expression used to represent a special
       data node, action, or notification instance identifier
       string.
```

A node-instance-identifier value is an unrestricted YANG instance-identifier expression. All the same rules as an instance-identifier apply except predicates for keys are optional. If a key predicate is missing, then the node-instance-identifier represents all possible server instances for that key.

This XPath expression is evaluated in the following context:

- o The set of namespace declarations are those in scope on the leaf element where this type is used.

- o The set of variable bindings contains one variable, 'USER', which contains the name of the user of the current session.
- o The function library is the core function library, but note that due to the syntax restrictions of an instance-identifier, no functions are allowed.
- o The context node is the root node in the data tree.

The accessible tree includes actions and notifications tied to data nodes.";

}

/\*

\* Data definition statements

\*/

```
container nacm {
  nacm:default-deny-all;

  description
    "Parameters for NETCONF Access Control Model.";

  leaf enable-nacm {
    type boolean;
    default true;
    description
      "Enables or disables all NETCONF access control
       enforcement.  If 'true', then enforcement
       is enabled.  If 'false', then enforcement
       is disabled.";
  }

  leaf read-default {
    type action-type;
    default "permit";
    description
      "Controls whether read access is granted if
       no appropriate rule is found for a
       particular read request.";
  }

  leaf write-default {
    type action-type;
    default "deny";
    description
      "Controls whether create, update, or delete access
```

```
        is granted if no appropriate rule is found for a
        particular write request.";
    }

    leaf exec-default {
        type action-type;
        default "permit";
        description
            "Controls whether exec access is granted if no appropriate
            rule is found for a particular protocol operation request.";
    }

    leaf enable-external-groups {
        type boolean;
        default true;
        description
            "Controls whether the server uses the groups reported by the
            NETCONF transport layer when it assigns the user to a set of
            NACM groups.  If this leaf has the value 'false', any group
            names reported by the transport layer are ignored by the
            server.";
    }

    leaf denied-operations {
        type yang:zero-based-counter32;
        config false;
        mandatory true;
        description
            "Number of times since the server last restarted that a
            protocol operation request was denied.";
    }

    leaf denied-data-writes {
        type yang:zero-based-counter32;
        config false;
        mandatory true;
        description
            "Number of times since the server last restarted that a
            protocol operation request to alter
            a configuration datastore was denied.";
    }

    leaf denied-notifications {
        type yang:zero-based-counter32;
        config false;
        mandatory true;
        description
            "Number of times since the server last restarted that
```

```
        a notification was dropped for a subscription because
        access to the event type was denied.";
    }

    container groups {
        description
            "NETCONF Access Control Groups.";

        list group {
            key name;

            description
                "One NACM Group Entry. This list will only contain
                configured entries, not any entries learned from
                any transport protocols.";

            leaf name {
                type group-name-type;
                description
                    "Group name associated with this entry.";
            }

            leaf-list user-name {
                type user-name-type;
                description
                    "Each entry identifies the username of
                    a member of the group associated with
                    this entry.";
            }
        }
    }

    list rule-list {
        key "name";
        ordered-by user;
        description
            "An ordered collection of access control rules.";

        leaf name {
            type string {
                length "1..max";
            }
            description
                "Arbitrary name assigned to the rule-list.";
        }
        leaf-list group {
            type union {
                type matchall-string-type;
            }
        }
    }
}
```

```
    type group-name-type;
  }
  description
    "List of administrative groups that will be
    assigned the associated access rights
    defined by the 'rule' list.

    The string '*' indicates that all groups apply to the
    entry.";
}

list rule {
  key "name";
  ordered-by user;
  description
    "One access control rule.

    Rules are processed in user-defined order until a match is
    found. A rule matches if 'module-name', 'rule-type', and
    'access-operations' match the request. If a rule
    matches, the 'action' leaf determines if access is granted
    or not.";

  leaf name {
    type string {
      length "1..max";
    }
    description
      "Arbitrary name assigned to the rule.";
  }

  leaf module-name {
    type union {
      type matchall-string-type;
      type string;
    }
    default "*";
    description
      "Name of the module associated with this rule.

      This leaf matches if it has the value '*' or if the
      object being accessed is defined in the module with the
      specified module name.";
  }

  choice rule-type {
    description
      "This choice matches if all leafs present in the rule
      match the request. If no leafs are present, the
```

```

        choice matches all requests.";
    case protocol-operation {
        leaf rpc-name {
            type union {
                type matchall-string-type;
                type string;
            }
            description
                "This leaf matches if it has the value '*' or if
                its value equals the requested protocol operation
                name.";
        }
    }
    case notification {
        leaf notification-name {
            type union {
                type matchall-string-type;
                type string;
            }
            description
                "This leaf matches if it has the value '*' or if its
                value equals the requested notification name.";
        }
    }
    case data-node {
        leaf path {
            type node-instance-identifier;
            mandatory true;
            description
                "Data Node Instance Identifier associated with the
                data node, action, or notification controlled by
                this rule.

                Configuration data or state data instance
                identifiers start with a top-level data node. A
                complete instance identifier is required for this
                type of path value.

                The special value '/' refers to all possible
                datastore contents.";
        }
    }
}

leaf access-operations {
    type union {
        type matchall-string-type;
        type access-operations-type;
    }
}

```

```
    }
    default "*";
    description
        "Access operations associated with this rule.

        This leaf matches if it has the value '*' or if the
        bit corresponding to the requested operation is set.";
}

leaf action {
    type action-type;
    mandatory true;
    description
        "The access control action associated with the
        rule.  If a rule is determined to match a
        particular request, then this object is used
        to determine whether to permit or deny the
        request.";
}

leaf comment {
    type string;
    description
        "A textual description of the access rule.";
}
}
}
}
}

<CODE ENDS>
```

Figure 5

### 3.6. IANA Considerations

This document reuses the URI for "ietf-netconf-acm" in "The IETF XML Registry".

This document updates the module registration in the "YANG Module Names" registry to reference this RFC instead of RFC 6536. Following the format in [RFC6020], the following has been registered.

```
Name: ietf-netconf-acm
Namespace: urn:ietf:params:xml:ns:yang:ietf-netconf-acm
Prefix: nacm
reference: RFC XXXX
```



### 3.7. Security Considerations

The YANG module defined in this document is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC5246].

The NETCONF access control model [RFCXXXX] 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 is a risk related to the lack of access control enforcement for the RESTCONF OPTIONS and PATCH methods. The risk here is that the response to OPTIONS and PATCH may vary based on the presence or absence of a resource corresponding to the URL's path. If this is the case, then it can be used to trivially probe for the presence or absence of values within a tree. Therefore, a server MUST NOT vary its responses based on the existence of the underlying resource, which would indicate the presence or absence of resource instances. In particular servers should not expose any instance information before ensuring that the client has the necessary access permissions to obtain that information. In such cases, servers are expected to always return the "access-denied" error response.

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:

- o /nacm : the entire /nacm subtree is related to security. Refer to the following sections for more details.

This section highlights the issues for an administrator to consider when configuring a NETCONF server with NACM.

#### 3.7.1. NACM Configuration and Monitoring Considerations

Configuration of the access control system is highly sensitive to system security. A server may choose not to allow any user configuration to some portions of it, such as the global security level or the groups that allowed access to system resources.

By default, NACM enforcement is enabled. By default, "read" access to all datastore contents is enabled (unless "nacm:default-deny-all" is specified for the data definition), and "exec" access is enabled for safe protocol operations. An administrator needs to ensure that NACM is enabled and also decide if the default access parameters are set appropriately. Make sure the following data nodes are properly configured:

- o /nacm/enable-nacm (default "true")
- o /nacm/read-default (default "permit")
- o /nacm/write-default (default "deny")
- o /nacm/exec-default (default "permit")

An administrator needs to restrict write access to all configurable objects within this data model.

If write access is allowed for configuration of access control rules, then care needs to be taken not to disrupt the access control enforcement. For example, if the NACM access control rules are edited directly within the running configuration datastore (i.e., :writable-running capability is supported and used), then care needs to be taken not to allow unintended access while the edits are being done.

An administrator needs to make sure that the translation from a transport- or implementation-dependent user identity to a NACM username is unique and correct. This requirement is specified in detail in Section 2.2 of [RFC6241].

An administrator needs to be aware that the YANG data structures representing access control rules (/nacm/rule-list and /nacm/rule-list/rule) are ordered by the client. The server will evaluate the access control rules according to their relative conceptual order within the running configuration datastore.

Note that the /nacm/groups data structure contains the administrative group names used by the server. These group names may be configured locally and/or provided through an external protocol, such as RADIUS [RFC2865][RFC5607].

An administrator needs to be aware of the security properties of any external protocol used by the transport layer to determine group names. For example, if this protocol does not protect against man-in-the-middle attacks, an attacker might be able to inject group names that are configured in NACM, so that a user gets more

permissions than it should. In such cases, the administrator may wish to disable the usage of such group names, by setting `/nacm/enable-external-groups` to "false".

An administrator needs to restrict read access to the following objects within this data model, as they reveal access control configuration that could be considered sensitive.

- o `/nacm/enable-nacm`
- o `/nacm/read-default`
- o `/nacm/write-default`
- o `/nacm/exec-default`
- o `/nacm/enable-external-groups`
- o `/nacm/groups`
- o `/nacm/rule-list`

### 3.7.2. General Configuration Issues

There is a risk that invocation of non-standard protocol operations will have undocumented side effects. An administrator needs to construct access control rules such that the configuration datastore is protected from such side effects.

It is possible for a session with some write access (e.g., allowed to invoke `<edit-config>`), but without any access to a particular datastore subtree containing sensitive data, to determine the presence or non-presence of that data. This can be done by repeatedly issuing some sort of edit request (create, update, or delete) and possibly receiving "access-denied" errors in response. These "fishing" attacks can identify the presence or non-presence of specific sensitive data even without the "error-path" field being present within the `<rpc-error>` response.

It may be possible for the set of NETCONF capabilities on the server to change over time. If so, then there is a risk that new protocol operations, notifications, and/or datastore content have been added to the device. An administrator needs to be sure the access control rules are correct for the new content in this case. Mechanisms to detect NETCONF capability changes on a specific device are outside the scope of this document.

It is possible that the data model definition itself (e.g., YANG when-stmt) will help an unauthorized session determine the presence or even value of sensitive data nodes by examining the presence and values of different data nodes.

It is possible that the data model definition itself (e.g., YANG when-stmt or choice-stmt) will allow a session to implicitly create or delete nodes that the session does not have write access to as an implicit side effect from the processing of an allowed <edit-config> operation.

There is a risk that non-standard protocol operations, or even the standard <get> protocol operation, may return data that "aliases" or "copies" sensitive data from a different data object. There may simply be multiple data model definitions that expose or even configure the same underlying system instrumentation.

A data model may contain external keys (e.g., YANG leafref), which expose values from a different data structure. An administrator needs to be aware of sensitive data models that contain leafref nodes. This entails finding all the leafref objects that "point" at the sensitive data (i.e., "path-stmt" values) that implicitly or explicitly include the sensitive data node.

It is beyond the scope of this document to define access control enforcement procedures for underlying device instrumentation that may exist to support the NETCONF server operation. An administrator can identify each protocol operation that the server provides and decide if it needs any access control applied to it.

This document incorporates the optional use of a recovery session mechanism, which can be used to bypass access control enforcement in emergencies, such as NACM configuration errors that disable all access to the server. The configuration and identification of such a recovery session mechanism are implementation-specific and outside the scope of this document. An administrator needs to be aware of any recovery session mechanisms available on the device and make sure they are used appropriately.

It is possible for a session to disrupt configuration management, even without any write access to the configuration, by locking the datastore. This may be done to ensure all or part of the configuration remains stable while it is being retrieved, or it may be done as a "denial-of-service" attack. There is no way for the server to know the difference. An administrator may wish to restrict "exec" access to the following protocol operations:

- o <lock>

- o <unlock>
- o <partial-lock>
- o <partial-unlock>

### 3.7.3. Data Model Design Considerations

Designers need to clearly identify any sensitive data, notifications, or protocol operations defined within a YANG module. For such definitions, a "nacm:default-deny-write" or "nacm:default-deny-all" statement ought to be present, in addition to a clear description of the security risks.

Protocol operations need to be properly documented by the data model designer, so it is clear to administrators what data nodes (if any) are affected by the protocol operation and what information (if any) is returned in the <rpc-reply> message.

Data models ought to be designed so that different access levels for input parameters to protocol operations are not required. Use of generic protocol operations should be avoided, and if different access levels are needed, separate protocol operations should be defined instead.

## 4. References

### 4.1. Normative References

- [I-D.ietf-netmod-revised-datastores]  
Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture", draft-ietf-netmod-revised-datastores-02 (work in progress), May 2017.
- [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>>.
- [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>>.
- [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>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

#### 4.2. Informative References

- [RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, DOI 10.17487/RFC2865, June 2000, <<https://www.rfc-editor.org/info/rfc2865>>.
- [RFC5607] Nelson, D. and G. Weber, "Remote Authentication Dial-In User Service (RADIUS) Authorization for Network Access Server (NAS) Management", RFC 5607, DOI 10.17487/RFC5607, July 2009, <<https://www.rfc-editor.org/info/rfc5607>>.

## Appendix A. Change Log

-- RFC Ed.: remove this section before publication.

The NACM issue tracker can be found here: <https://github.com/netconf-wg/rfc6536bis/issues>

## A.1. v08 to v09

- o Clarify that a data rule applies to a data node and its descendants.
- o Clarify that edit side effects are not subject to access control.

## A.2. v07 to v08

- o Address IESG review comments

## A.3. v06 to v07

- o Clarify CRUDX protocol operation mapping

## A.4. v05 to v06

- o Change title to remove the word NETCONF
- o Clarify data rule case in YANG module
- o Update security considerations section

## A.5. v04 to v05

- o Clarify NETCONF protocol operation vs added operation via additional YANG modules
- o Change term 'NETCONF transport layer' to 'transport layer'
- o Clarify that read access operations are not coupled to specific protocol operations
- o Update date of YANG module so it matches new revision date

## A.6. v03 to v04

- o Fix revision date mismatch for extracting YANG module

## A.7. v02 to v03

- o Clarify NACM YANG extensions for use outside NACM

## A.8. v01 to v02

- o Corrected section title for changes since RFC 6536.
- o Added section 'Mapping New Datastores to NACM'.
- o Changed term NETCONF datastore to datastore/
- o Removed text about RESTCONF and a conceptual datastore.

## A.9. v00 to v01

- o Updated RESTCONF reference

## A.10. v00

- o Renamed document from draft-bierman-netconf-rfc6536bis-01 to draft-ietf-netconf-rfc6536bis-00.

## Appendix B. Usage Examples

The following XML snippets are provided as examples only, to demonstrate how NACM can be configured to perform some access control tasks.

## B.1. &lt;groups&gt; Example

There needs to be at least one <group> entry in order for any of the access control rules to be useful.

The following XML shows arbitrary groups and is not intended to represent any particular use case.



```
<nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
  <groups>
    <group>
      <name>admin</name>
      <user-name>admin</user-name>
      <user-name>andy</user-name>
    </group>

    <group>
      <name>limited</name>
      <user-name>wilma</user-name>
      <user-name>bam-bam</user-name>
    </group>

    <group>
      <name>guest</name>
      <user-name>guest</user-name>
      <user-name>guest@example.com</user-name>
    </group>
  </groups>
</nacm>
```

This example shows three groups:

admin: The "admin" group contains two users named "admin" and "andy".

limited: The "limited" group contains two users named "wilma" and "bam-bam".

guest: The "guest" group contains two users named "guest" and "guest@example.com".

## B.2. Module Rule Example

Module rules are used to control access to all the content defined in a specific module. A module rule has the `<module-name>` leaf set, but no case in the "rule-type" choice.

```
<nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
  <rule-list>
    <name>guest-acl</name>
    <group>guest</group>

  </rule>
```

```
<name>deny-ncm</name>
<module-name>ietf-netconf-monitoring</module-name>
<access-operations>*</access-operations>
<action>deny</action>
<comment>
    Do not allow guests any access to the NETCONF
    monitoring information.
</comment>
</rule>
</rule-list>

<rule-list>
  <name>limited-acl</name>
  <group>limited</group>

  <rule>
    <name>permit-ncm</name>
    <module-name>ietf-netconf-monitoring</module-name>
    <access-operations>read</access-operations>
    <action>permit</action>
    <comment>
        Allow read access to the NETCONF
        monitoring information.
    </comment>
  </rule>
  <rule>
    <name>permit-exec</name>
    <module-name>*</module-name>
    <access-operations>exec</access-operations>
    <action>permit</action>
    <comment>
        Allow invocation of the
        supported server operations.
    </comment>
  </rule>
</rule-list>

<rule-list>
  <name>admin-acl</name>
  <group>admin</group>

  <rule>
    <name>permit-all</name>
    <module-name>*</module-name>
    <access-operations>*</access-operations>
    <action>permit</action>
    <comment>
        Allow the admin group complete access to all
```

```
        operations and data.  
    </comment>  
</rule>  
</rule-list>  
</nacm>
```

This example shows four module rules:

deny-nacm: This rule prevents the "guest" group from reading any monitoring information in the "ietf-netconf-monitoring" YANG module.

permit-nacm: This rule allows the "limited" group to read the "ietf-netconf-monitoring" YANG module.

permit-exec: This rule allows the "limited" group to invoke any protocol operation supported by the server.

permit-all: This rule allows the "admin" group complete access to all content in the server. No subsequent rule will match for the "admin" group because of this module rule.

### B.3. Protocol Operation Rule Example

Protocol operation rules are used to control access to a specific protocol operation.

```
<nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">  
  <rule-list>  
    <name>guest-limited-acl</name>  
    <group>limited</group>  
    <group>guest</group>  
  
    <rule>  
      <name>deny-kill-session</name>  
      <module-name>ietf-netconf</module-name>  
      <rpc-name>kill-session</rpc-name>  
      <access-operations>exec</access-operations>  
      <action>deny</action>  
      <comment>  
        Do not allow the limited or guest group  
        to kill another session.  
      </comment>  
    </rule>  
  </rule>
```

```
<name>deny-delete-config</name>
<module-name>ietf-netconf</module-name>
<rpc-name>delete-config</rpc-name>
<access-operations>exec</access-operations>
<action>deny</action>
<comment>
  Do not allow limited or guest group
  to delete any configurations.
</comment>
</rule>
</rule-list>

<rule-list>
  <name>limited-acl</name>
  <group>limited</group>

  <rule>
    <name>permit-edit-config</name>
    <module-name>ietf-netconf</module-name>
    <rpc-name>edit-config</rpc-name>
    <access-operations>exec</access-operations>
    <action>permit</action>
    <comment>
      Allow the limited group to edit the configuration.
    </comment>
  </rule>
</rule-list>

</nacm>
```

This example shows three protocol operation rules:

**deny-kill-session:** This rule prevents the "limited" or "guest" groups from invoking the NETCONF <kill-session> protocol operation.

**deny-delete-config:** This rule prevents the "limited" or "guest" groups from invoking the NETCONF <delete-config> protocol operation.

**permit-edit-config:** This rule allows the "limited" group to invoke the NETCONF <edit-config> protocol operation. This rule will have no real effect unless the "exec-default" leaf is set to "deny".

#### B.4. Data Node Rule Example

Data node rules are used to control access to specific (config and non-config) data nodes within the NETCONF content provided by the server.

```
<nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
  <rule-list>
    <name>guest-acl</name>
    <group>guest</group>

    <rule>
      <name>deny-nacm</name>
      <path xmlns:n="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
        /n:nacm
      </path>
      <access-operations>*</access-operations>
      <action>deny</action>
      <comment>
        Deny the guest group any access to the /nacm data.
      </comment>
    </rule>
  </rule-list>

  <rule-list>
    <name>limited-acl</name>
    <group>limited</group>

    <rule>
      <name>permit-acme-config</name>
      <path xmlns:acme="http://example.com/ns/netconf">
        /acme:acme-netconf/acme:config-parameters
      </path>
      <access-operations>
        read create update delete
      </access-operations>
      <action>permit</action>
      <comment>
        Allow the limited group complete access to the acme
        NETCONF configuration parameters. Showing long form
        of 'access-operations' instead of shorthand.
      </comment>
    </rule>
  </rule-list>

  <rule-list>
    <name>guest-limited-acl</name>
    <group>guest</group>
```

```
<group>limited</group>

<rule>
  <name>permit-dummy-interface</name>
  <path xmlns:acme="http://example.com/ns/itf">
    /acme:interfaces/acme:interface[acme:name='dummy']
  </path>
  <access-operations>read update</access-operations>
  <action>permit</action>
  <comment>
    Allow the limited and guest groups read
    and update access to the dummy interface.
  </comment>
</rule>
</rule-list>

<rule-list>
  <name>admin-acl</name>
  <group>admin</group>
  <rule>
    <name>permit-interface</name>
    <path xmlns:acme="http://example.com/ns/itf">
      /acme:interfaces/acme:interface
    </path>
    <access-operations>*</access-operations>
    <action>permit</action>
    <comment>
      Allow admin full access to all acme interfaces.
    </comment>
  </rule>
</rule-list>
</nacm>
```

This example shows four data node rules:

deny-nacm: This rule denies the "guest" group any access to the <nacm> subtree.

permit-acme-config: This rule gives the "limited" group read-write access to the acme <config-parameters>.

permit-dummy-interface: This rule gives the "limited" and "guest" groups read-update access to the acme <interface> entry named "dummy". This entry cannot be created or deleted by these groups, just altered.

permit-interface: This rule gives the "admin" group read-write access to all acme <interface> entries.

#### B.5. Notification Rule Example

Notification rules are used to control access to a specific notification event type.

```
<nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
  <rule-list>
    <name>sys-acl</name>
    <group>limited</group>
    <group>guest</group>

    <rule>
      <name>deny-config-change</name>
      <module-name>acme-system</module-name>
      <notification-name>sys-config-change</notification-name>
      <access-operations>read</access-operations>
      <action>deny</action>
      <comment>
        Do not allow the guest or limited groups
        to receive config change events.
      </comment>
    </rule>
  </rule-list>
</nacm>
```

This example shows one notification rule:

deny-config-change: This rule prevents the "limited" or "guest" groups from receiving the acme <sys-config-change> event type.

#### Authors' Addresses

Andy Bierman  
YumaWorks  
685 Cochran St.  
Suite #160  
Simi Valley, CA 93065  
USA

EMail: andy@yumaworks.com

Martin Bjorklund  
Tail-f Systems

EMail: [mbj@tail-f.com](mailto:mbj@tail-f.com)



NETCONF Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: September 14, 2017

K. Watsen  
Juniper Networks  
G. Wu  
Cisco Systems  
March 13, 2017

SSH Client and Server Models  
draft-ietf-netconf-ssh-client-server-02

Abstract

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

Editorial Note (To be removed by RFC Editor)

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

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

- o I-D.ietf-netconf-keystore

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

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

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

- o "2017-03-13" --> the publication date of this draft

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

- o Appendix A. Change Log

- o Appendix B. Open Issues

## 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 <http://datatracker.ietf.org/drafts/current/>.

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

This Internet-Draft will expire on September 14, 2017.

## Copyright Notice

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

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

## Table of Contents

1. Introduction . . . . .	3
1.1. Terminology . . . . .	3
1.2. Tree Diagrams . . . . .	4
2. The SSH Client Model . . . . .	4
2.1. Tree Diagram . . . . .	4
2.2. Example Usage . . . . .	5
2.3. YANG Model . . . . .	6
3. The SSH Server Model . . . . .	10
3.1. Tree Diagram . . . . .	10
3.2. Example Usage . . . . .	11
3.3. YANG Model . . . . .	12
4. The SSH Common Model . . . . .	15

4.1. Tree Diagram . . . . .	16
4.2. Example Usage . . . . .	16
4.3. YANG Model . . . . .	17
5. Security Considerations . . . . .	27
6. IANA Considerations . . . . .	28
6.1. The IETF XML Registry . . . . .	28
6.2. The YANG Module Names Registry . . . . .	29
7. Acknowledgements . . . . .	29
8. References . . . . .	29
8.1. Normative References . . . . .	29
8.2. Informative References . . . . .	30
Appendix A. Change Log . . . . .	32
A.1. server-model-09 to 00 . . . . .	32
A.2. 00 to 01 . . . . .	32
A.3. 01 to 02 . . . . .	32
Appendix B. Open Issues . . . . .	32
Authors' Addresses . . . . .	32

## 1. Introduction

This document defines three YANG [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 (SSH is defined in [RFC4252], [RFC4253], and [RFC4254]). It is intended that these groupings will be used by applications using the SSH protocol. 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 enables 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 grouping for the SSH parts it provides, while adding data nodes for the TCP-level call-home configuration.

### 1.1. Terminology

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

## 1.2. Tree Diagrams

A simplified graphical representation of the data models is used in this document. The meaning of the symbols in these diagrams is as follows:

- o Brackets "[" and "]" enclose list keys.
- o Braces "{" and "}" enclose feature names, and indicate that the named feature must be present for the subtree to be present.
- o Abbreviations before data node names: "rw" means configuration (read-write) and "ro" state data (read-only).
- o Symbols after data node names: "?" means an optional node, "!" means a presence container, and "\*" denotes a list and leaf-list.
- o Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- o Ellipsis ("...") stands for contents of subtrees that are not shown.

## 2. The SSH Client Model

The SSH client model presented in this section contains one YANG grouping, to just configure the SSH client omitting, for instance, any configuration for which IP address or port the client should connect to.

This grouping references data nodes defined by the keystore model [I-D.ietf-netconf-keystore]. For instance, a reference to the keystore model is made to indicate which trusted CA certificate a client should use to authenticate X.509v3 certificate based host keys [RFC6187].

### 2.1. Tree Diagram

The following tree diagram presents the data model for the grouping defined in the ietf-ssh-client module. Please see Section 1.2 for tree diagram notation.

```

module: ietf-ssh-client
groupings:
ssh-client-grouping
  +----- server-auth
  |   +----- trusted-ssh-host-keys?
  |   |       -> /ks:keystore/trusted-host-keys/name
  |   +----- trusted-ca-certs?
  |   |       -> /ks:keystore/trusted-certificates/name
  |   |       {sshcom:ssh-x509-certs}?
  |   +----- trusted-server-certs?
  |   |       -> /ks:keystore/trusted-certificates/name
  |   |       {sshcom:ssh-x509-certs}?
  +----- client-auth
  |   +----- username?          string
  |   +----- (auth-type)?
  |   |   +---:(certificate)
  |   |   |   +----- certificate?  leafref {sshcom:ssh-x509-certs}?
  |   |   +---:(public-key)
  |   |   |   +----- public-key?   -> /ks:keystore/keys/key/name
  |   |   +---:(password)
  |   |   |   +----- password?     union
  +----- transport-params {ssh-client-transport-params-config}?
  |   +----- host-key
  |   |   +----- host-key-alg*    identityref
  |   +----- key-exchange
  |   |   +----- key-exchange-alg* identityref
  |   +----- encryption
  |   |   +----- encryption-alg*  identityref
  |   +----- mac
  |   |   +----- mac-alg*         identityref
  |   +----- compression
  |   |   +----- compression-alg* identityref

```

## 2.2. Example Usage

This section shows how it would appear if the `ssh-client-grouping` were populated with some data. This example is consistent with the examples presented in Section 2.2 of [I-D.ietf-netconf-keystore].

```
<!-- hypothetical example, as groupings don't have instance data -->
<ssh-client xmlns="urn:ietf:params:xml:ns:yang:ietf-ssh-client">

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

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

### 2.3. YANG Model

This YANG module has a normative references to [RFC6991] and [I-D.ietf-netconf-keystore].

<CODE BEGINS> file "ietf-ssh-client@2017-03-13.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 sshcom;
    revision-date 2017-03-13; // stable grouping definitions
    reference
      "RFC XXXX: SSH Client and Server Models";
  }

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

  import ietf-keystore {
    prefix ks;
    reference
      "RFC YYYY: Keystore Model";
```

```
}

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

contact
  "WG Web:    <http://tools.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) 2014 IETF Trust and the persons identified as
  authors of the code. All rights reserved.

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

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

revision "2017-03-13" {
  description
    "Initial version";
  reference
    "RFC XXXX: SSH Client and Server Models";
}

feature ssh-client-transport-params-config {
  description
    "SSH transport layer parameters are configurable on an SSH
    client.";
}

grouping ssh-client-grouping {
  description
```

"A reusable grouping for configuring a SSH client without any consideration for how an underlying TCP session is established.";

```
container server-auth {
  description
    "Trusted server identities.";

  leaf trusted-ssh-host-keys {
    type leafref {
      path "/ks:keystore/ks:trusted-host-keys/ks:name";
    }
    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 authenticate if it is an exact
       match to a configured trusted SSH host key.";
  }

  leaf trusted-ca-certs {
    if-feature sshcom:ssh-x509-certs;
    type leafref {
      path "/ks:keystore/ks:trusted-certificates/ks:name";
    }
    description
      "A reference to a list of certificate authority (CA)
       certificates used by the SSH client to authenticate
       SSH server certificates.";
  }

  leaf trusted-server-certs {
    if-feature sshcom:ssh-x509-certs;
    type leafref {
      path "/ks:keystore/ks:trusted-certificates/ks:name";
    }
    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 trusted server certificate.";
  }
}

container client-auth {
  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 {
  description
    "The authentication type.";
  leaf certificate {
    if-feature sshcom:ssh-x509-certs;
    type leafref {
      path "/ks:keystore/ks:keys/ks:key/ks:certificates/"
        + "ks:certificate/ks:name";
    }
    description
      "A certificates to be used for user authentication.";
  }
  leaf public-key {
    type leafref {
      path "/ks:keystore/ks:keys/ks:key/ks:name";
    }
    description
      "A public keys to be used for user authentication.";
  }
  leaf password {
    nacm:default-deny-all;
    type union {
      type string;
      type enumeration {
        enum "RESTRICTED" {
          description
            "The private key is restricted due to access-control.";
        }
      }
    }
    description
      "A password to be used for user authentication.";
  }
}
} // end client-auth

container transport-params {
  if-feature ssh-client-transport-params-config;
  uses sshcom:transport-params-grouping;
  description
    "Configurable parameters for the SSH transport layer.";
```

```
    }  
  } // ssh-client-grouping  
}
```

<CODE ENDS>

### 3. The SSH Server Model

The SSH server model presented in this section contains one YANG grouping, for just the SSH-level configuration omitting, for instance, configuration for which ports to open to listen for connections on.

This grouping references data nodes defined by the keystore model [I-D.ietf-netconf-keystore]. For instance, a reference to the keystore model is made to indicate which host key a server should present.

#### 3.1. Tree Diagram

The following tree diagram presents the data model for the grouping defined in the ietf-ssh-server module. Please see Section 1.2 for tree diagram notation.

```

module: ietf-ssh-server
groupings:
ssh-server-grouping
+----- host-keys
|   +----- host-key* [name]
|   |   +----- name?          string
|   |   +----- (host-key-type)
|   |   |   +---:(public-key)
|   |   |   |   +----- public-key?    -> /ks:keystore/keys/key/name
|   |   |   |   +---:(certificate)
|   |   |   |   +----- certificate?    leafref {sshcom:ssh-x509-certs}?
+----- client-cert-auth {sshcom:ssh-x509-certs}?
|   +----- trusted-ca-certs?
|   |   -> /ks:keystore/trusted-certificates/name
+----- trusted-client-certs?
|   -> /ks:keystore/trusted-certificates/name
+----- transport-params {ssh-server-transport-params-config}?
|   +----- host-key
|   |   +----- host-key-alg*    identityref
+----- key-exchange
|   +----- key-exchange-alg*    identityref
+----- encryption
|   +----- encryption-alg*      identityref
+----- mac
|   +----- mac-alg*             identityref
+----- compression
|   +----- compression-alg*     identityref

```

### 3.2. Example Usage

This section shows how it would appear if the ssh-server-grouping were populated with some data. This example is consistent with the examples presented in Section 2.2 of [I-D.ietf-netconf-keystore].

```

<!-- hypothetical example, as groupings don't have instance data -->
<ssh-server xmlns="urn:ietf:params:xml:ns:yang:ietf-ssh-server">

  <!-- which host-keys will this SSH server present -->
  <host-keys>
    <host-key>
      <name>deployment-specific-certificate</name>
      <certificate>ex-rsa-cert</certificate>
    </host-key>
  </host-keys>

  <!-- NOTE: password/public-key auth is NOT configured here, -->
  <!-- as it is configured in the ietf-system (RFC 7317) -->
  <!-- module instead. -->

  <!-- which client-certs will this SSH server trust -->
  <client-cert-auth>
    <trusted-ca-certs>deployment-specific-ca-certs</trusted-ca-certs>
    <trusted-client-certs>explicitly-trusted-client-certs</trusted-client-certs>
  </client-cert-auth>

</ssh-server>

```

### 3.3. YANG Model

This YANG module has a normative references to [RFC4253], [RFC6991], and [I-D.ietf-netconf-keystore].

```

<CODE BEGINS> file "ietf-ssh-server@2017-03-13.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 sshcom;
    revision-date 2017-03-13; // stable grouping definitions
    reference
      "RFC XXXX: SSH Client and Server Models";
  }

  import ietf-keystore {
    prefix ks;
    reference
      "RFC YYYY: Keystore Model";
  }

```

```
}

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

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

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

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) 2014 IETF Trust and the persons identified as
  authors of the code. All rights reserved.

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

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

revision "2017-03-13" {
  description
    "Initial version";
  reference
    "RFC XXXX: SSH Client and Server Models";
}

// features
feature ssh-server-transport-params-config {
  description
    "SSH transport layer parameters are configurable on an SSH
    server.";
}

// grouping
grouping ssh-server-grouping {
  description
    "A reusable grouping for configuring a SSH server without
```

```

    any consideration for how underlying TCP sessions are
    established.";
container host-keys {
  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";
      leaf public-key {
        type leafref {
          path "/ks:keystore/ks:keys/ks:key/ks:name";
        }
        description
          "The public key is actually identified by the name of
          its cooresponding private-key in the keystore.";
      }
      leaf certificate {
        if-feature sshcom:ssh-x509-certs;
        type leafref {
          path "/ks:keystore/ks:keys/ks:key/ks:certificates/"
            + "ks:certificate/ks:name";
        }
        description
          "The name of a certificate in the keystore.";
      }
    }
  }
}
}

container client-cert-auth {

```

```
    if-feature sshcom:ssh-x509-certs;
    description
        "A reference to a list of trusted certificate authority (CA)
        certificates and a reference to a list of trusted client
        certificates.";
    leaf trusted-ca-certs {
        type leafref {
            path "/ks:keystore/ks:trusted-certificates/ks:name";
        }
        description
            "A reference to a list of certificate authority (CA)
            certificates used by the SSH server to authenticate
            SSH client certificates.";
    }

    leaf trusted-client-certs {
        type leafref {
            path "/ks:keystore/ks:trusted-certificates/ks:name";
        }
        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 trusted client certificate.";
    }
}

container transport-params {
    if-feature ssh-server-transport-params-config;
    uses sshcom:transport-params-grouping;
    description
        "Configurable parameters for the SSH transport layer.";
}

} // ssh-server-grouping

}
```

<CODE ENDS>

#### 4. 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.

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.

#### 4.1. Tree Diagram

The following tree diagram presents the data model for the grouping defined in the ietf-ssh-common module. Please see Section 1.2 for tree diagram notation.

```
module: ietf-ssh-common
  groupings:
    transport-params-grouping
      +---- host-key
      |   +---- host-key-alg*   identityref
      +---- key-exchange
      |   +---- key-exchange-alg*   identityref
      +---- encryption
      |   +---- encryption-alg*   identityref
      +---- mac
      |   +---- mac-alg*   identityref
      +---- compression
      |   +---- compression-alg*   identityref
```

#### 4.2. Example Usage

This section shows how it would appear if the transport-params-grouping were populated with some data.



```
<!-- hypothetical example, as groupings don't have instance data -->
<transport-params xmlns="urn:ietf:params:xml:ns:yang:ietf-ssh-common">

  <host-key>
    <host-key-alg>x509v3-rsa2048-sha256</host-key-alg>
    <host-key-alg>ssh-rsa</host-key-alg>
  </host-key>
  <key-exchange>
    <key-exchange-alg>
      diffie-hellman-group-exchange-sha256
    </key-exchange-alg>
  </key-exchange>
  <encryption>
    <encryption-alg>aes256-ctr</encryption-alg>
    <encryption-alg>aes192-ctr</encryption-alg>
    <encryption-alg>aes128-ctr</encryption-alg>
    <encryption-alg>aes256-cbc</encryption-alg>
    <encryption-alg>aes192-cbc</encryption-alg>
    <encryption-alg>aes128-cbc</encryption-alg>
  </encryption>
  <mac>
    <mac-alg>hmac-sha2-256</mac-alg>
    <mac-alg>hmac-sha2-512</mac-alg>
  </mac>
  <compression>
    <compression-alg>none</compression-alg>
  </compression>

</transport-params>
```

#### 4.3. YANG Model

This YANG module has a normative references to [RFC4344], [RFC4419], and [RFC5656].

```
<CODE BEGINS> file "ietf-ssh-common@2017-03-13.yang"

module ietf-ssh-common {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-ssh-common";
  prefix "sshcom";

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

  contact
```

"WG Web: <<http://tools.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) 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 "2017-03-13" {  
  description  
    "Initial version";  
  reference  
    "RFC XXXX: SSH Client and Server Models";  
}  
  
// 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 as 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)";
    }

    feature ssh-zlib {
        description
            "ZLIB (LZ77) compression is supported for SSH.";
        reference
            "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
    }

    // 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";
    }

    identity compression-alg-base {
        description
            "Base identity used to identify compression algorithms.";
    }

    identity none {
        base compression-alg-base;
        description
            "No compression.";
        reference
            "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
    }

    identity zlib {
        base compression-alg-base;
        if-feature ssh-zlib;
        description
```

```
    "ZLIB (LZ77) compression.";
  reference
    "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
}

// groupings
grouping transport-params-grouping {
  description
    "A reusable grouping for SSH transport parameters.
    For configurable parameters, a zero-element leaf-list of
    algorithms indicates the system default configuration for that
    parameter.";
  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
        "Host key algorithms in order of descending preference.";
    }
  }
  container key-exchange {
    description
      "Parameters regarding key exchange.";
    leaf-list key-exchange-alg {
      type identityref {
        base key-exchange-alg-base;
      }
      ordered-by user;
      description
        "Key exchange algorithms in order of descending
        preference.";
    }
  }
  container encryption {
    description
      "Parameters regarding encryption.";
    leaf-list encryption-alg {
      type identityref {
        base encryption-alg-base;
      }
      ordered-by user;
      description

```

```
        "Encryption algorithms in order of descending preference.";
    }
}
container mac {
    description
        "Parameters regarding message authentication code (MAC).";
    leaf-list mac-alg {
        type identityref {
            base mac-alg-base;
        }
        ordered-by user;
        description
            "MAC algorithms in order of descending preference.";
    }
}
container compression {
    description
        "Parameters regarding compression.";
    leaf-list compression-alg {
        type identityref {
            base compression-alg-base;
        }
        ordered-by user;
        description
            "Compression algorithms in order of descending preference.";
    }
}
}
```

<CODE ENDS>

## 5. 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) [RFC6536] 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 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:

/client-auth/password: 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. This being the case, this node is marked with the NACM value 'default-deny-all'.

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

NONE

## 6. IANA Considerations

### 6.1. The IETF XML Registry

This document registers three URIs in 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.

## 6.2. The YANG Module Names Registry

This document registers three YANG modules in the YANG Module Names registry [RFC7950]. Following the format in [RFC7950], 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:	sshcom
reference:	RFC XXXX

## 7. Acknowledgements

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

## 8. References

### 8.1. Normative References

[I-D.ietf-netconf-keystore]  
Watsen, K. and G. Wu, "Keystore Model", draft-ietf-netconf-keystore-00 (work in progress), October 2016.

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC4344] Bellare, M., Kohno, T., and C. Namprempre, "The Secure Shell (SSH) Transport Layer Encryption Modes", RFC 4344, DOI 10.17487/RFC4344, January 2006, <<http://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, <<http://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, <<http://www.rfc-editor.org/info/rfc5656>>.
- [RFC6187] Igoe, K. and D. Stebila, "X.509v3 Certificates for Secure Shell Authentication", RFC 6187, DOI 10.17487/RFC6187, March 2011, <<http://www.rfc-editor.org/info/rfc6187>>.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <<http://www.rfc-editor.org/info/rfc6536>>.
- [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, <<http://www.rfc-editor.org/info/rfc6668>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<http://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, <<http://www.rfc-editor.org/info/rfc7950>>.

## 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, <<http://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, <<http://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, <<http://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, <<http://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, <<http://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, <<http://www.rfc-editor.org/info/rfc6242>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<http://www.rfc-editor.org/info/rfc8040>>.
- [RFC8071] Watsen, K., "NETCONF Call Home and RESTCONF Call Home", RFC 8071, DOI 10.17487/RFC8071, February 2017, <<http://www.rfc-editor.org/info/rfc8071>>.

## Appendix A. Change Log

## A.1. server-model-09 to 00

- o This draft was split out from draft-ietf-netconf-server-model-09.
- o Added in previously missing ietf-ssh-client module.
- o Noted that '0.0.0.0' and ':::' might have special meanings.

## A.2. 00 to 01

- o Renamed "keychain" to "keystore".

## A.3. 01 to 02

- o Removed the groupings 'listening-ssh-client-grouping' and 'listening-ssh-server-grouping'. Now modules only contain the transport-independent groupings.
- o Simplified the "client-auth" part in the ietf-ssh-client module. It now inlines what it used to point to keystore for.
- o Added cipher suites for various algorithms into new 'ietf-ssh-common' module.

## Appendix B. Open Issues

Please see: <https://github.com/netconf-wg/ssh-client-server/issues>.

## Authors' Addresses

Kent Watsen  
Juniper Networks

EMail: [kwatsen@juniper.net](mailto:kwatsen@juniper.net)

Gary Wu  
Cisco Systems

EMail: [garywu@cisco.com](mailto:garywu@cisco.com)



NETCONF Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: 8 September 2022

K. Watsen  
Watsen Networks  
7 March 2022

YANG Groupings for SSH Clients and SSH Servers  
draft-ietf-netconf-ssh-client-server-27

Abstract

This document defines three YANG 1.1 modules: the first defines features and groupings common to both SSH clients and SSH servers, the second defines a grouping for a generic SSH client, and the third defines a grouping for a generic SSH server.

Editorial Note (To be removed by RFC Editor)

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

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

- \* AAAA --> the assigned RFC value for draft-ietf-netconf-crypto-types
- \* BBBB --> the assigned RFC value for draft-ietf-netconf-trust-anchors
- \* CCCC --> the assigned RFC value for draft-ietf-netconf-keystore
- \* DDDD --> the assigned RFC value for draft-ietf-netconf-tcp-client-server
- \* EEEE --> 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:

- \* 2022-03-07 --> the publication date of this draft

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

- \* Appendix B. Change Log

## Status of This Memo

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

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

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

This Internet-Draft will expire on 8 September 2022.

## Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

1. Introduction . . . . .	4
1.1. Relation to other RFCs . . . . .	5
1.2. Specification Language . . . . .	6
1.3. Adherence to the NMDA . . . . .	6
1.4. Conventions . . . . .	6
2. The "ietf-ssh-common" Module . . . . .	7
2.1. Data Model Overview . . . . .	7
2.2. Example Usage . . . . .	8
2.3. YANG Module . . . . .	9
3. The "ietf-ssh-client" Module . . . . .	15
3.1. Data Model Overview . . . . .	15
3.2. Example Usage . . . . .	17
3.3. YANG Module . . . . .	21
4. The "ietf-ssh-server" Module . . . . .	28
4.1. Data Model Overview . . . . .	28

4.2.	Example Usage . . . . .	31
4.3.	YANG Module . . . . .	35
5.	Security Considerations . . . . .	44
5.1.	The "iana-ssh-key-exchange-algs" Module . . . . .	44
5.2.	The "iana-ssh-encryption-algs" Module . . . . .	44
5.3.	The "iana-ssh-mac-algs" Module . . . . .	45
5.4.	The "iana-ssh-public-key-algs" Module . . . . .	45
5.5.	The "ietf-ssh-common" YANG Module . . . . .	46
5.6.	The "ietf-ssh-client" YANG Module . . . . .	47
5.7.	The "ietf-ssh-server" YANG Module . . . . .	48
6.	IANA Considerations . . . . .	48
6.1.	The "IETF XML" Registry . . . . .	48
6.2.	The "YANG Module Names" Registry . . . . .	49
6.3.	The "iana-ssh-encryption-algs" Module . . . . .	50
6.4.	The "iana-ssh-mac-algs" Module . . . . .	51
6.5.	The "iana-ssh-public-key-algs" Module . . . . .	51
6.6.	The "iana-ssh-key-exchange-algs" Module . . . . .	52
7.	References . . . . .	52
7.1.	Normative References . . . . .	52
7.2.	Informative References . . . . .	54
Appendix A.	YANG Modules for IANA . . . . .	56
A.1.	Initial Module for the "Encryption Algorithm Names" Registry . . . . .	56
A.1.1.	Data Model Overview . . . . .	57
A.1.2.	Example Usage . . . . .	58
A.1.3.	YANG Module . . . . .	58
A.2.	Initial Module for the "MAC Algorithm Names" Registry . .	66
A.2.1.	Data Model Overview . . . . .	66
A.2.2.	Example Usage . . . . .	67
A.2.3.	YANG Module . . . . .	68
A.3.	Initial Module for the "Public Key Algorithm Names" Registry . . . . .	71
A.3.1.	Data Model Overview . . . . .	71
A.3.2.	Example Usage . . . . .	73
A.3.3.	YANG Module . . . . .	73
A.4.	Initial Module for the "Key Exchange Method Names" Registry . . . . .	82
A.4.1.	Data Model Overview . . . . .	82
A.4.2.	Example Usage . . . . .	84
A.4.3.	YANG Module . . . . .	84
Appendix B.	Change Log . . . . .	130
B.1.	00 to 01 . . . . .	130
B.2.	01 to 02 . . . . .	131
B.3.	02 to 03 . . . . .	131
B.4.	03 to 04 . . . . .	131
B.5.	04 to 05 . . . . .	132
B.6.	05 to 06 . . . . .	132
B.7.	06 to 07 . . . . .	132

B.8.	07 to 08	132
B.9.	08 to 09	132
B.10.	09 to 10	133
B.11.	10 to 11	133
B.12.	11 to 12	133
B.13.	12 to 13	133
B.14.	13 to 14	133
B.15.	14 to 15	133
B.16.	15 to 16	134
B.17.	16 to 17	134
B.18.	17 to 18	134
B.19.	18 to 19	135
B.20.	19 to 20	135
B.21.	20 to 21	135
B.22.	21 to 22	136
B.23.	22 to 23	136
B.24.	23 to 24	136
B.25.	24 to 25	136
B.26.	25 to 26	137
B.27.	26 to 27	137
Acknowledgements		137
Contributors		137
Author's Address		137

## 1. Introduction

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

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

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

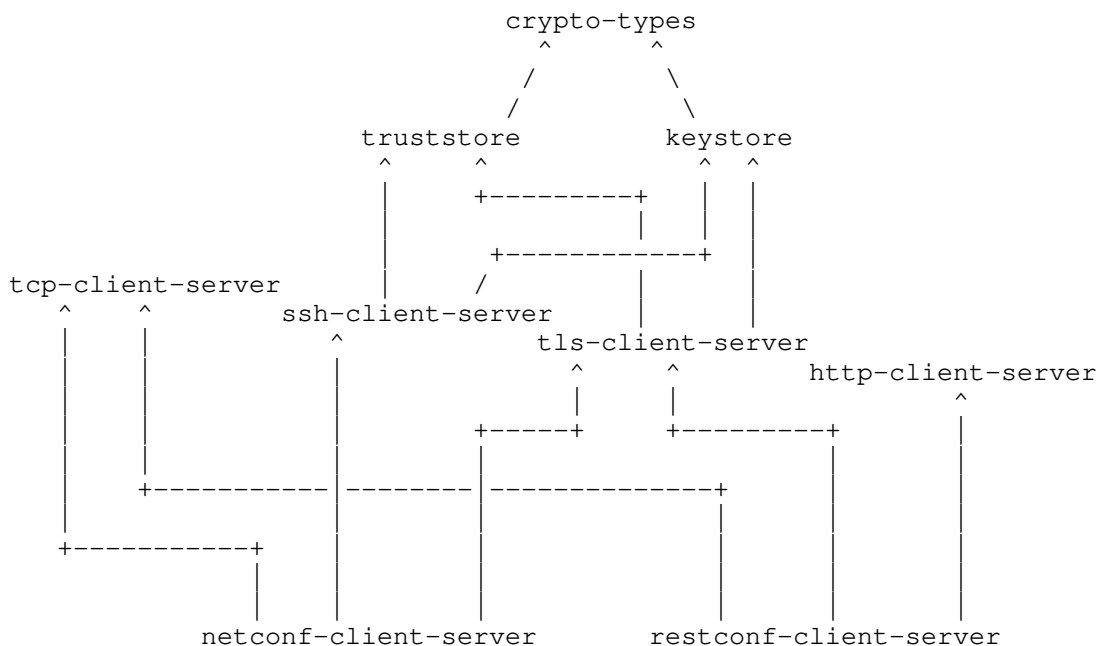
The modules defined in this document optionally support [RFC6187] enabling X.509v3 certificate based host keys and public keys.

### 1.1. Relation to other RFCs

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

The modules have been defined in a modular fashion to enable their use by other efforts, some of which are known to be in progress at the time of this writing, with many more expected to be defined in time.

The normative dependency relationship between the various RFCs in the collection is presented in the below diagram. The labels in the diagram represent the primary purpose provided by each RFC. Hyperlinks to each RFC are provided below the diagram.



Label in Diagram	Originating RFC
crypto-types	[I-D.ietf-netconf-crypto-types]
truststore	[I-D.ietf-netconf-trust-anchors]
keystore	[I-D.ietf-netconf-keystore]
tcp-client-server	[I-D.ietf-netconf-tcp-client-server]
ssh-client-server	[I-D.ietf-netconf-ssh-client-server]
tls-client-server	[I-D.ietf-netconf-tls-client-server]
http-client-server	[I-D.ietf-netconf-http-client-server]
netconf-client-server	[I-D.ietf-netconf-netconf-client-server]
restconf-client-server	[I-D.ietf-netconf-restconf-client-server]

Table 1: Label to RFC Mapping

## 1.2. Specification Language

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

## 1.3. Adherence to the NMDA

This document is compliant with the Network Management Datastore Architecture (NMDA) [RFC8342]. For instance, as described in [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore], trust anchors and keys installed during manufacturing are expected to appear in <operational>.

## 1.4. Conventions

Various examples used in this document use a placeholder value for binary data that has been base64 encoded (e.g., "BASE64VALUE="). This placeholder value is used as real base64 encoded structures are often many lines long and hence distracting to the example being presented.

## 2. The "ietf-ssh-common" Module

The SSH common model presented in this section contains features and groupings common to both SSH clients and SSH servers. The "transport-params-grouping" grouping can be used to configure the list of SSH transport algorithms permitted by the SSH client or SSH server. The lists of permitted algorithms are in decreasing order of usage preference. The algorithm that appears first in the client list that also appears in the server list is the one that is used for the SSH transport layer connection. The ability to restrict the algorithms allowed is provided in this grouping for SSH clients and SSH servers that are capable of doing so and may serve to make SSH clients and SSH servers compliant with security policies.

### 2.1. Data Model Overview

This section provides an overview of the "ietf-ssh-common" module in terms of its features, identities, and groupings.

#### 2.1.1. Features

The following diagram lists all the "feature" statements defined in the "ietf-ssh-common" module:

Features:

```
+-- ssh-x509-certs
+-- transport-params
```

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

#### 2.1.2. Groupings

The "ietf-ssh-common" module defines the following "grouping" statement:

```
* transport-params-grouping
```

This grouping is presented in the following subsection.

##### 2.1.2.1. The "transport-params-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "transport-params-grouping" grouping:

```
grouping transport-params-grouping
  +-- host-key
  |   +-- host-key-alg*   identityref
  +-- key-exchange
  |   +-- key-exchange-alg*   identityref
  +-- encryption
  |   +-- encryption-alg*   identityref
  +-- mac
      +-- mac-alg*   identityref
```

Comments:

- \* This grouping is used by both the "ssh-client-grouping" and the "ssh-server-grouping" groupings defined in Section 3.1.2.1 and Section 4.1.2.1, respectively.
- \* This grouping enables client and server configurations to specify the algorithms that are to be used when establishing SSH sessions.
- \* Each list is "ordered-by user".

### 2.1.3. Protocol-accessible Nodes

The "ietf-ssh-common" module defines only "grouping" statements that are used by other modules to instantiate protocol-accessible nodes.

### 2.2. Example Usage

This following example illustrates how the "transport-params-grouping" grouping appears when populated with some data.



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

<!-- The outermost element below doesn't exist in the data model. -->  
<!-- It simulates if the "grouping" were a "container" instead. -->

```
<transport-params
  xmlns="urn:ietf:params:xml:ns:yang:ietf-ssh-common"
  xmlns:sshea="urn:ietf:params:xml:ns:yang:iana-ssh-encryption-algs"
  xmlns:sshkea="urn:ietf:params:xml:ns:yang:iana-ssh-key-exchange-al\
gs"
  xmlns:sshma="urn:ietf:params:xml:ns:yang:iana-ssh-mac-algs"
  xmlns:sshpkc="urn:ietf:params:xml:ns:yang:iana-ssh-public-key-algs"
">
  <host-key>
    <host-key-alg>sshpkc:x509v3-rsa2048-sha256</host-key-alg>
    <host-key-alg>sshpkc:ssh-rsa</host-key-alg>
  </host-key>
  <key-exchange>
    <key-exchange-alg>
      sshkea:diffie-hellman-group-exchange-sha256
    </key-exchange-alg>
  </key-exchange>
  <encryption>
    <encryption-alg>sshea:aes256-ctr</encryption-alg>
    <encryption-alg>sshea:aes192-ctr</encryption-alg>
    <encryption-alg>sshea:aes128-ctr</encryption-alg>
    <encryption-alg>sshea:aes256-cbc</encryption-alg>
    <encryption-alg>sshea:aes192-cbc</encryption-alg>
    <encryption-alg>sshea:aes128-cbc</encryption-alg>
  </encryption>
  <mac>
    <mac-alg>sshma:hmac-sha2-256</mac-alg>
    <mac-alg>sshma:hmac-sha2-512</mac-alg>
  </mac>
</transport-params>
```

### 2.3. YANG Module

This YANG module has normative references to [RFC4253], [RFC4344], [RFC4419], [RFC5656], [RFC6187], and [RFC6668].

<CODE BEGINS> file "ietf-ssh-common@2022-03-07.yang"

```
module ietf-ssh-common {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-ssh-common";
  prefix sshcmn;

  import iana-ssh-encryption-algs {
    prefix sshea;
    reference
      "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
  }

  import iana-ssh-key-exchange-algs {
    prefix sshkea;
    reference
      "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
  }

  import iana-ssh-mac-algs {
    prefix sshma;
    reference
      "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
  }

  import iana-ssh-public-key-algs {
    prefix sshpka;
    reference
      "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
  }

  import ietf-crypto-types {
    prefix ct;
    reference
      "RFC AAAA: YANG Data Types and Groupings for Cryptography";
  }

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

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

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

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

description

"This module defines a common features and groupings for Secure Shell (SSH).

Copyright (c) 2021 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

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

```
revision 2022-03-07 {
  description
    "Initial version";
  reference
    "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
}
```

// Features

```
feature ssh-x509-certs {
  description
    "X.509v3 certificates are supported for SSH.";
  reference
    "RFC 6187: X.509v3 Certificates for Secure Shell
      Authentication";
}

feature transport-params {
  description
    "SSH transport layer parameters are configurable.";
}
```

```
feature public-key-generation {
  description
    "Indicates that the server implements the
     'generate-public-key' RPC.";
}

// 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 sshpka: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 EEEE for
         valid combinations.

         If this leaf-list is not configured (has zero elements)
         the acceptable host key algorithms are implementation-
         defined.";
      reference
        "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
    }
  }
  container key-exchange {
    description
      "Parameters regarding key exchange.";
    leaf-list key-exchange-alg {
      type identityref {
        base sshkea: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 sshea: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 sshma: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.";
    }
}
}

// Protocol-accessible Nodes

rpc generate-public-key {
    if-feature "public-key-generation";
    description
        "Requests the device to generate an public key using
        the specified key algorithm.";
    input {
```

```
leaf algorithm {
  type sshpka:public-key-algorithm-ref;
  mandatory true;
  description
    "The algorithm to be used when generating the key.";
}
leaf bits {
  type uint16;
  description
    "Specifies the number of bits in the key to create.
    For RSA keys, the minimum size is 1024 bits and
    the default is 3072 bits. Generally, 3072 bits is
    considered sufficient. DSA keys must be exactly 1024
    bits as specified by FIPS 186-2. For ECDSA keys, the
    'bits' value determines the key length by selecting
    from one of three elliptic curve sizes: 256, 384 or
    521 bits. Attempting to use bit lengths other than
    these three values for ECDSA keys will fail. ECDSA-SK,
    Ed25519 and Ed25519-SK keys have a fixed length and
    the 'bits' value, if specified, will be ignored.";
}
choice private-key-encoding {
  default cleartext;
  description
    "A choice amongst optional private key handling.";
  case cleartext {
    leaf cleartext {
      type empty;
      description
        "Indicates that the private key is to be returned
        as a cleartext value.";
    }
  }
  case encrypt {
    if-feature "ct:private-key-encryption";
    container encrypt-with {
      description
        "Indicates that the key is to be encrypted using
        the specified symmetric or asymmetric key.";
      uses ks:encrypted-by-choice-grouping;
    }
  }
  case hide {
    if-feature "ct:hidden-keys";
    leaf hide {
      type empty;
      description
        "Indicates that the private key is to be hidden.
```

```
        Unlike the 'cleartext' and 'encrypt' options, the
        key returned is a placeholder for an internally
        stored key. See the 'Support for Built-in Keys'
        section in RFC CCCC for information about hidden
        keys.";
    }
}
}
output {
    uses ct:asymmetric-key-pair-grouping;
}
} // end generate-public-key

}

<CODE ENDS>
```

### 3. The "ietf-ssh-client" Module

This section defines a YANG 1.1 [RFC7950] module called "ietf-ssh-client". A high-level overview of the module is provided in Section 3.1. Examples illustrating the module's use are provided in Examples (Section 3.2). The YANG module itself is defined in Section 3.3.

#### 3.1. Data Model Overview

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

##### 3.1.1. Features

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

Features:

```
+-- ssh-client-keepalives
+-- client-ident-password
+-- client-ident-publickey
+-- client-ident-hostbased
+-- client-ident-none
```

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

### 3.1.2. Groupings

The "ietf-ssh-client" module defines the following "grouping" statement:

\* ssh-client-grouping

This grouping is presented in the following subsection.

#### 3.1.2.1. The "ssh-client-grouping" Grouping

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

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

```

grouping ssh-client-grouping
+-- client-identity
|   +-- username?      string
|   +-- public-key! {client-ident-publickey}?
|   |   +---u ks:local-or-keystore-asymmetric-key-grouping
|   +-- password! {client-ident-password}?
|   |   +---u ct:password-grouping
|   +-- hostbased! {client-ident-hostbased}?
|   |   +---u ks:local-or-keystore-asymmetric-key-grouping
|   +-- none?          empty {client-ident-none}?
|   +-- certificate! {sshcmn:ssh-x509-certs}?
|       +---u ks:local-or-keystore-end-entity-cert-with-key-grouping
ng
+-- server-authentication
|   +-- ssh-host-keys!
|   |   +---u ts:local-or-truststore-public-keys-grouping
|   +-- ca-certs! {sshcmn:ssh-x509-certs}?
|   |   +---u ts:local-or-truststore-certs-grouping
|   +-- ee-certs! {sshcmn:ssh-x509-certs}?
|       +---u ts:local-or-truststore-certs-grouping
+-- transport-params {sshcmn:transport-params}?
|   +---u sshcmn:transport-params-grouping
+-- keepalives! {ssh-client-keepalives}?
    +-- max-wait?      uint16
    +-- max-attempts?  uint8

```

Comments:

- \* The "client-identity" node configures a "username" and authentication methods, each enabled by a "feature" statement defined in Section 3.1.1.



- \* The "server-authentication" node configures trust anchors for authenticating the SSH server, with each option enabled by a "feature" statement.
- \* The "transport-params" node, which must be enabled by a feature, configures parameters for the SSH sessions established by this configuration.
- \* The "keepalives" node, which must be enabled by a feature, configures a "presence" container for testing the aliveness of the SSH server. The aliveness-test occurs at the SSH protocol layer.
- \* For the referenced grouping statement(s):
  - The "local-or-keystore-asymmetric-key-grouping" grouping is discussed in Section 2.1.3.4 of [I-D.ietf-netconf-keystore].
  - The "local-or-keystore-end-entity-cert-with-key-grouping" grouping is discussed in Section 2.1.3.6 of [I-D.ietf-netconf-keystore].
  - The "local-or-truststore-public-keys-grouping" grouping is discussed in Section 2.1.3.2 of [I-D.ietf-netconf-trust-anchors].
  - The "local-or-truststore-certs-grouping" grouping is discussed in Section 2.1.3.1 of [I-D.ietf-netconf-trust-anchors].
  - The "transport-params-grouping" grouping is discussed in Section 2.1.2.1 in this document.

### 3.1.3. Protocol-accessible Nodes

The "ietf-ssh-client" module defines only "grouping" statements that are used by other modules to instantiate protocol-accessible nodes.

### 3.2. Example Usage

This section presents two examples showing the "ssh-client-grouping" grouping populated with some data. These examples are effectively the same except the first configures the client identity using a local key while the second uses a key configured in a keystore. Both examples are consistent with the examples presented in Section 2 of [I-D.ietf-netconf-trust-anchors] and Section 3.2 of [I-D.ietf-netconf-keystore].

The following configuration example uses local-definitions for the client identity and server authentication:

```

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

<!-- The outermost element below doesn't exist in the data model. -->
<!-- It simulates if the "grouping" were a "container" instead. -->

<ssh-client
  xmlns="urn:ietf:params:xml:ns:yang:ietf-ssh-client"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types"
  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>
      <local-definition>
        <public-key-format>ct:ssh-public-key-format</public-key-form\
at>
        <public-key>BASE64VALUE=</public-key>
        <private-key-format>ct:rsa-private-key-format</private-key-f\
ormat>
        <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
      </local-definition>
    </public-key>
  </client-identity>

  <!-- which host keys will this client trust -->
  <server-authentication>
    <ssh-host-keys>
      <local-definition>
        <public-key>
          <name>corp-fw1</name>
          <public-key-format>ct:ssh-public-key-format</public-key-fo\
rmat>
          <public-key>BASE64VALUE=</public-key>
        </public-key>
        <public-key>
          <name>corp-fw2</name>
          <public-key-format>ct:ssh-public-key-format</public-key-fo\
rmat>
          <public-key>BASE64VALUE=</public-key>
        </public-key>
      </local-definition>
    </ssh-host-keys>
    <ca-certs>
      <local-definition>
        <certificate>
          <name>Server Cert Issuer #1</name>
          <cert-data>BASE64VALUE=</cert-data>

```

```
        </certificate>
      <certificate>
        <name>Server Cert Issuer #2</name>
        <cert-data>BASE64VALUE=</cert-data>
      </certificate>
    </local-definition>
  </ca-certs>
  <ee-certs>
    <local-definition>
      <certificate>
        <name>My Application #1</name>
        <cert-data>BASE64VALUE=</cert-data>
      </certificate>
      <certificate>
        <name>My Application #2</name>
        <cert-data>BASE64VALUE=</cert-data>
      </certificate>
    </local-definition>
  </ee-certs>
</server-authentication>

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

</ssh-client>
```

The following configuration example uses keystore-references for the client identity and truststore-references for server authentication: from the keystore:

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

<!-- The outermost element below doesn't exist in the data model. -->  
 <!-- It simulates if the "grouping" were a "container" instead. -->

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

  <!-- how this client will authenticate itself to the server -->
  <client-identity>
    <username>foobar</username>
    <!-- can an SSH client have more than one key?
    <public-key>
      <keystore-reference>ssh-rsa-key</keystore-reference>
    </public-key>
    -->
    <certificate>
      <keystore-reference>
        <asymmetric-key>ssh-rsa-key-with-cert</asymmetric-key>
        <certificate>ex-rsa-cert2</certificate>
      </keystore-reference>
    </certificate>
  </client-identity>

  <!-- which host-keys will this client trust -->
  <server-authentication>
    <ssh-host-keys>
      <truststore-reference>trusted-ssh-public-keys</truststore-refe\
rence>
    </ssh-host-keys>
    <ca-certs>
      <truststore-reference>trusted-server-ca-certs</truststore-refe\
rence>
    </ca-certs>
    <ee-certs>
      <truststore-reference>trusted-server-ee-certs</truststore-refe\
rence>
    </ee-certs>
  </server-authentication>

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

</ssh-client>
```

### 3.3. YANG Module

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

<CODE BEGINS> file "ietf-ssh-client@2022-03-07.yang"

```
module ietf-ssh-client {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-ssh-client";
  prefix sshc;

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

  import ietf-crypto-types {
    prefix ct;
    reference
      "RFC AAAA: YANG Data Types and Groupings for Cryptography";
  }

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

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

  import ietf-ssh-common {
    prefix sshcmn;
    revision-date 2022-03-07; // stable grouping definitions
    reference
      "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
  }

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

  contact
    "WG Web:  https://datatracker.ietf.org/wg/netconf
```

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

description

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

Copyright (c) 2021 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

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

```
revision 2022-03-07 {  
  description  
    "Initial version";  
  reference  
    "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";  
}
```

// Features

```
feature ssh-client-keepalives {  
  description  
    "Per socket SSH keepalive parameters are configurable for  
    SSH clients on the server implementing this feature.";  
}
```

```
feature client-ident-publickey {  
  description  
    "Indicates that the 'publickey' authentication type, per  
    RFC 4252, is supported for client identification."
```

```
    The 'publickey' authentication type is required by
    RFC 4252, but common implementations enable it to
    be disabled.";
reference
  "RFC 4252:
    The Secure Shell (SSH) Authentication Protocol";
}

feature client-ident-password {
  description
    "Indicates that the 'password' authentication type, per
    RFC 4252, is supported for client identification.";
  reference
    "RFC 4252:
      The Secure Shell (SSH) Authentication Protocol";
}

feature client-ident-hostbased {
  description
    "Indicates that the 'hostbased' authentication type, per
    RFC 4252, is supported for client identification.";
  reference
    "RFC 4252:
      The Secure Shell (SSH) Authentication Protocol";
}

feature client-ident-none {
  description
    "Indicates that the 'none' authentication type, per
    RFC 4252, is supported for client identification.";
  reference
    "RFC 4252:
      The Secure Shell (SSH) Authentication Protocol";
}

// Groupings

grouping ssh-client-grouping {
  description
    "A reusable grouping for configuring a SSH client without
    any consideration for how an underlying TCP session is
    established.

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

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

```
container client-identity {
  nacm:default-deny-write;
  description
    "The username and authentication methods for the client.
    The authentication methods are unordered. Clients may
    initially send any configured method or, per RFC 4252,
    Section 5.2, send the 'none' method to prompt the server
    to provide a list of productive methods. Whenever a
    choice amongst methods arises, implementations SHOULD
    use a default ordering that prioritizes automation
    over human-interaction.";
  leaf username {
    type string;
    description
      "The username of this user. This will be the username
      used, for instance, to log into an SSH server.";
  }
  container public-key {
    if-feature "client-ident-publickey";
    presence
      "Indicates that publickey-based authentication has been
      configured. This statement is present so the mandatory
      descendant nodes do not imply that this node must be
      configured.";
    description
      "A locally-defined or referenced asymmetric key
      pair to be used for client identification.";
    reference
      "RFC CCCC: A YANG Data Model for a Keystore";
    uses ks:local-or-keystore-asymmetric-key-grouping {
      refine "local-or-keystore/local/local-definition" {
        must 'public-key-format = "ct:ssh-public-key-format"';
      }
      refine "local-or-keystore/keystore/keystore-reference" {
        must 'deref(..)/../ks:public-key-format'
          + ' = "ct:ssh-public-key-format"';
      }
    }
  }
}
container password {
  if-feature "client-ident-password";
  presence
    "Indicates that password-based authentication has been
    configured. This statement is present so the mandatory
```



```
        descendant nodes do not imply that this node must be
        configured.";
    description
        "A password to be used to authenticate the client's
        identity.";
    uses ct:password-grouping;
}
container hostbased {
    if-feature "client-ident-hostbased";
    presence
        "Indicates that hostbased authentication is configured.
        This statement is present so the mandatory descendant
        nodes do not imply that this node must be configured.";
    description
        "A locally-defined or referenced asymmetric key
        pair to be used for host identification.";
    reference
        "RFC CCCC: A YANG Data Model for a Keystore";
    uses ks:local-or-keystore-asymmetric-key-grouping {
        refine "local-or-keystore/local/local-definition" {
            must 'public-key-format = "ct:ssh-public-key-format"';
        }
        refine "local-or-keystore/keystore/keystore-reference" {
            must 'deref(..)/../ks:public-key-format'
                + ' = "ct:ssh-public-key-format"';
        }
    }
}
leaf none {
    if-feature "client-ident-none";
    type empty;
    description
        "Indicates that 'none' algorithm is used for client
        identification.";
}
container certificate {
    if-feature "sshcmn:ssh-x509-certs";
    presence
        "Indicates that certificate-based authentication has been
        configured. This statement is present so the mandatory
        descendant nodes do not imply that this node must be
        configured.";
    description
        "A locally-defined or referenced certificate
        to be used for client identification.";
    reference
        "RFC CCCC: A YANG Data Model for a Keystore";
    uses ks:local-or-keystore-end-entity-cert-with-key-grouping {
```

```
    refine "local-or-keystore/local/local-definition" {
      must 'public-key-format'
      + ' = "ct:subject-public-key-info-format"';
    }
    refine "local-or-keystore/keystore/keystore-reference"
      + "/asymmetric-key" {
      must 'deref(..)/../ks:public-key-format'
      + ' = "ct:subject-public-key-info-format"';
    }
  }
} // container client-identity

container server-authentication {
  nacm:default-deny-write;
  must 'ssh-host-keys or ca-certs or ee-certs';
  description
    "Specifies how the SSH client can authenticate SSH servers.
    Any combination of authentication methods is additive and
    unordered.";
  container ssh-host-keys {
    presence
      "Indicates that the SSH host key have been configured.
      This statement is present so the mandatory descendant
      nodes do not imply that this node must be configured.";
    description
      "A bag 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 BBBB: A YANG Data Model for a Truststore";
    uses ts:local-or-truststore-public-keys-grouping {
      refine
        "local-or-truststore/local/local-definition/public-key" {
          must 'public-key-format = "ct:ssh-public-key-format"';
        }
      refine
        "local-or-truststore/truststore/truststore-reference" {
          must 'deref(..)/../*/ts:public-key-format'
          + ' = "ct:ssh-public-key-format"';
        }
    }
  }
}

container ca-certs {
  if-feature "sshcmn:ssh-x509-certs";
  presence
    "Indicates that the CA certificates have been configured."
```

```
        This statement is present so the mandatory descendant
        nodes do not imply that this node must be configured.";
    description
        "A set of certificate authority (CA) certificates used by
        the SSH client to authenticate SSH servers. A server
        is authenticated if its certificate has a valid chain
        of trust to a configured CA certificate.";
    reference
        "RFC BBBB: A YANG Data Model for a Truststore";
    uses ts:local-or-truststore-certs-grouping;
}
container ee-certs {
    if-feature "sshcmn:ssh-x509-certs";
    presence
        "Indicates that the EE certificates have been configured.
        This statement is present so the mandatory descendant
        nodes do not imply that this node must be configured.";
    description
        "A set of end-entity certificates used by the SSH client
        to authenticate SSH servers. A server is authenticated
        if its certificate is an exact match to a configured
        end-entity certificate.";
    reference
        "RFC BBBB: A YANG Data Model for a Truststore";
    uses ts:local-or-truststore-certs-grouping;
}
} // container server-authentication

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

container keepalives {
    nacm:default-deny-write;
    if-feature "ssh-client-keepalives";
    presence
        "Indicates that the SSH client proactively tests the
        aliveness of the remote SSH server.";
    description
        "Configures the keep-alive policy, to proactively test
        the aliveness of the SSH server. An unresponsive TLS
        server is dropped after approximately max-wait *
        max-attempts seconds. Per Section 4 of RFC 4254,
        the SSH client SHOULD send an SSH_MSG_GLOBAL_REQUEST
```

```
        message with a purposely nonexistent 'request name'
        value (e.g., keepalive@ietf.org) and the 'want reply'
        value set to '1'.";
    reference
        "RFC 4254: The Secure Shell (SSH) Connection Protocol";
    leaf max-wait {
        type uint16 {
            range "1..max";
        }
        units "seconds";
        default "30";
        description
            "Sets the amount of time in seconds after which if
            no data has been received from the SSH server, a
            TLS-level message will be sent to test the
            aliveness of the SSH server.";
    }
    leaf max-attempts {
        type uint8;
        default "3";
        description
            "Sets the maximum number of sequential keep-alive
            messages that can fail to obtain a response from
            the SSH server before assuming the SSH server is
            no longer alive.";
    }
} // container keepalives
} // grouping ssh-client-grouping
}
```

<CODE ENDS>

#### 4. The "ietf-ssh-server" Module

This section defines a YANG 1.1 module called "ietf-ssh-server". A high-level overview of the module is provided in Section 4.1. Examples illustrating the module's use are provided in Examples (Section 4.2). The YANG module itself is defined in Section 4.3.

##### 4.1. Data Model Overview

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

#### 4.1.1. Features

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

Features:

```
+-- ssh-server-keepalives
+-- local-users-supported
+-- local-user-auth-publickey {local-users-supported}?
+-- local-user-auth-password {local-users-supported}?
+-- local-user-auth-hostbased {local-users-supported}?
+-- local-user-auth-none {local-users-supported}?
```

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

#### 4.1.2. Groupings

The "ietf-ssh-server" module defines the following "grouping" statement:

```
* ssh-server-grouping
```

This grouping is presented in the following subsection.

##### 4.1.2.1. The "ssh-server-grouping" Grouping

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

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

```

grouping ssh-server-grouping
+-- server-identity
|   +-- host-key* [name]
|       +-- name?                string
|       +-- (host-key-type)
|           +--:(public-key)
|               +-- public-key
|                   +---u ks:local-or-keystore-asymmetric-key-grouping
|           +--:(certificate)
|               +-- certificate {sshcmn:ssh-x509-certs}?
|                   +---u ks:local-or-keystore-end-entity-cert-with-k\
ey-grouping
+-- client-authentication
|   +-- users {local-users-supported}?
|       +-- user* [name]
|           +-- name?            string
|           +-- public-keys! {local-user-auth-publickey}?
|               +---u ts:local-or-truststore-public-keys-grouping
|           +-- password?        ianach:crypt-hash
|               {local-user-auth-password}?
|           +-- hostbased! {local-user-auth-hostbased}?
|               +---u ts:local-or-truststore-public-keys-grouping
|           +-- none?            empty {local-user-auth-none}?
+-- ca-certs! {sshcmn:ssh-x509-certs}?
|   +---u ts:local-or-truststore-certs-grouping
+-- ee-certs! {sshcmn:ssh-x509-certs}?
|   +---u ts:local-or-truststore-certs-grouping
+-- transport-params {sshcmn:transport-params}?
|   +---u sshcmn:transport-params-grouping
+-- keepalives! {ssh-server-keepalives}?
|   +-- max-wait?                uint16
|   +-- max-attempts?           uint8

```

#### Comments:

- \* The "server-identity" node configures the authentication methods the server can use to identify itself to clients. The ability to use a certificate is enabled by a "feature".
- \* The "client-authentication" node configures trust anchors for authenticating the SSH client, with each option enabled by a "feature" statement.
- \* The "transport-params" node, which must be enabled by a feature, configures parameters for the SSH sessions established by this configuration.

- \* The "keepalives" node, which must be enabled by a feature, configures a "presence" container for testing the aliveness of the SSH client. The aliveness-test occurs at the SSH protocol layer.
- \* For the referenced grouping statement(s):
  - The "local-or-keystore-asymmetric-key-grouping" grouping is discussed in Section 2.1.3.4 of [I-D.ietf-netconf-keystore].
  - The "local-or-keystore-end-entity-cert-with-key-grouping" grouping is discussed in Section 2.1.3.6 of [I-D.ietf-netconf-keystore].
  - The "local-or-truststore-public-keys-grouping" grouping is discussed in Section 2.1.3.2 of [I-D.ietf-netconf-trust-anchors].
  - The "local-or-truststore-certs-grouping" grouping is discussed in Section 2.1.3.1 of [I-D.ietf-netconf-trust-anchors].
  - The "transport-params-grouping" grouping is discussed in Section 2.1.2.1 in this document.

#### 4.1.3. Protocol-accessible Nodes

The "ietf-ssh-server" module defines only "grouping" statements that are used by other modules to instantiate protocol-accessible nodes.

#### 4.2. Example Usage

This section presents two examples showing the "ssh-server-grouping" grouping populated with some data. These examples are effectively the same except the first configures the server identity using a local key while the second uses a key configured in a keystore. Both examples are consistent with the examples presented in Section 2 of [I-D.ietf-netconf-trust-anchors] and Section 3.2 of [I-D.ietf-netconf-keystore].

The following configuration example uses local-definitions for the server identity and client authentication:

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

```
<!-- The outermost element below doesn't exist in the data model. -->
<!-- It simulates if the "grouping" were a "container" instead. -->
```

```
<ssh-server
  xmlns="urn:ietf:params:xml:ns:yang:ietf-ssh-server"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types"
  xmlns:algs="urn:ietf:params:xml:ns:yang:ietf-ssh-common">
```

```
<!-- the host-key this SSH server will present -->
```

```

    <server-identity>
      <host-key>
        <name>my-pubkey-based-host-key</name>
        <public-key>
          <local-definition>
            <public-key-format>ct:ssh-public-key-format</public-key-fo\
rmat>
            <public-key>BASE64VALUE=</public-key>
            <private-key-format>ct:rsa-private-key-format</private-key\
-format>
            <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
          </local-definition>
        </public-key>
      </host-key>
      <host-key>
        <name>my-cert-based-host-key</name>
        <certificate>
          <local-definition>
            <public-key-format>ct:subject-public-key-info-format</publ\
ic-key-format>
            <public-key>BASE64VALUE=</public-key>
            <private-key-format>ct:rsa-private-key-format</private-key\
-format>
            <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
            <cert-data>BASE64VALUE=</cert-data>
          </local-definition>
        </certificate>
      </host-key>
    </server-identity>

    <!-- the client credentials this SSH server will trust -->
    <client-authentication>
      <users>
        <user>
          <name>mary</name>
          <password>$0$secret</password>
          <public-keys>
            <local-definition>
              <public-key>
                <name>User A</name>
                <public-key-format>ct:ssh-public-key-format</public-ke\
y-format>
                <public-key>BASE64VALUE=</public-key>
              </public-key>
              <public-key>
                <name>User B</name>
                <public-key-format>ct:ssh-public-key-format</public-ke\
y-format>

```



```
        <public-key>BASE64VALUE=</public-key>
      </public-key>
    </local-definition>
  </public-keys>
</user>
</users>
<ca-certs>
  <local-definition>
    <certificate>
      <name>Identity Cert Issuer #1</name>
      <cert-data>BASE64VALUE=</cert-data>
    </certificate>
    <certificate>
      <name>Identity Cert Issuer #2</name>
      <cert-data>BASE64VALUE=</cert-data>
    </certificate>
  </local-definition>
</ca-certs>
<ee-certs>
  <local-definition>
    <certificate>
      <name>Application #1</name>
      <cert-data>BASE64VALUE=</cert-data>
    </certificate>
    <certificate>
      <name>Application #2</name>
      <cert-data>BASE64VALUE=</cert-data>
    </certificate>
  </local-definition>
</ee-certs>
</client-authentication>

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

</ssh-server>
```

The following configuration example uses keystore-references for the server identity and truststore-references for client authentication: from the keystore:

```

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

<!-- The outermost element below doesn't exist in the data model. -->
<!-- It simulates if the "grouping" were a "container" instead. -->

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

  <!-- the host-key this SSH server will present -->
  <server-identity>
    <host-key>
      <name>my-pubkey-based-host-key</name>
      <public-key>
        <keystore-reference>ssh-rsa-key</keystore-reference>
      </public-key>
    </host-key>
    <host-key>
      <name>my-cert-based-host-key</name>
      <certificate>
        <keystore-reference>
          <asymmetric-key>ssh-rsa-key-with-cert</asymmetric-key>
          <certificate>ex-rsa-cert2</certificate>
        </keystore-reference>
      </certificate>
    </host-key>
  </server-identity>

  <!-- the client credentials this SSH server will trust -->
  <client-authentication>
    <users>
      <user>
        <name>mary</name>
        <password>$0$secret</password>
        <public-keys>
          <truststore-reference>SSH Public Keys for Application A</t\
ruststore-reference>
        </public-keys>
      </user>
    </users>
    <ca-certs>
      <truststore-reference>trusted-client-ca-certs</truststore-refe\
rence>
    </ca-certs>
    <ee-certs>
      <truststore-reference>trusted-client-ee-certs</truststore-refe\
rence>
    </ee-certs>
  </client-authentication>
</ssh-server>

```

```
</client-authentication>

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

</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@2022-03-07.yang"

```
module ietf-ssh-server {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-ssh-server";
  prefix sshs;

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

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

  import ietf-crypto-types {
    prefix ct;
    reference
      "RFC AAAA: YANG Data Types and Groupings for Cryptography";
  }

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

  import ietf-keystore {
    prefix ks;
```

```
reference
  "RFC CCCC: A YANG Data Model for a Keystore";
}

import ietf-ssh-common {
  prefix sshcmn;
  revision-date 2022-03-07; // stable grouping definitions
  reference
    "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
}

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

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

description
  "This module defines reusable groupings for SSH servers that
  can be used as a basis for specific SSH server instances.

  Copyright (c) 2021 IETF Trust and the persons identified
  as authors of the code. All rights reserved.

  Redistribution and use in source and binary forms, with
  or without modification, is permitted pursuant to, and
  subject to the license terms contained in, the Revised
  BSD License set forth in Section 4.c of the IETF Trust's
  Legal Provisions Relating to IETF Documents
  (https://trustee.ietf.org/license-info).https://www.rfc-editor.org/info/rfcEEEE); see the RFC
  itself for full legal notices.

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

revision 2022-03-07 {
  description
    "Initial version";
```

```
    reference
      "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
  }

  // Features

  feature ssh-server-keepalives {
    description
      "Per socket SSH keepalive parameters are configurable for
      SSH servers on the server implementing this feature.";
  }

  feature local-users-supported {
    description
      "Indicates that the configuration for users can be
      configured herein, as opposed to in an application
      specific location.";
  }

  feature local-user-auth-publickey {
    if-feature "local-users-supported";
    description
      "Indicates that the 'publickey' authentication type,
      per RFC 4252, is supported for locally-defined users.

      The 'publickey' authentication type is required by
      RFC 4252, but common implementations enable it to
      be disabled.";
    reference
      "RFC 4252:
      The Secure Shell (SSH) Authentication Protocol";
  }

  feature local-user-auth-password {
    if-feature "local-users-supported";
    description
      "Indicates that the 'password' authentication type,
      per RFC 4252, is supported for locally-defined users.";
    reference
      "RFC 4252:
      The Secure Shell (SSH) Authentication Protocol";
  }

  feature local-user-auth-hostbased {
    if-feature "local-users-supported";
    description
      "Indicates that the 'hostbased' authentication type,
      per RFC 4252, is supported for locally-defined users.";
```

```
reference
  "RFC 4252:
    The Secure Shell (SSH) Authentication Protocol";
}

feature local-user-auth-none {
  if-feature "local-users-supported";
  description
    "Indicates that the 'none' authentication type, per
    RFC 4252, is supported. It is NOT RECOMMENDED to
    enable this feature.";
  reference
    "RFC 4252:
      The Secure Shell (SSH) Authentication Protocol";
}

// Groupings

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

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

  container server-identity {
    nacm:default-deny-write;
    description
      "The list of host keys the SSH server will present when
      establishing a SSH connection.";
    list host-key {
      key "name";
      min-elements 1;
      ordered-by user;
      description
        "An ordered list of host keys the SSH server will use to
        construct its ordered list of algorithms, when sending
        its SSH_MSG_KEXINIT message, as defined in Section 7.1
        of RFC 4253.";
      reference
```

```
    "RFC 4253: The Secure Shell (SSH) Transport Layer
      Protocol";
  leaf name {
    type string;
    description
      "An arbitrary name for this host key";
  }
  choice host-key-type {
    mandatory true;
    description
      "The type of host key being specified";
    container public-key {
      description
        "A locally-defined or referenced asymmetric key pair
          to be used for the SSH server's host key.";
      reference
        "RFC CCCC: A YANG Data Model for a Keystore";
      uses ks:local-or-keystore-asymmetric-key-grouping {
        refine "local-or-keystore/local/local-definition" {
          must
            'public-key-format = "ct:ssh-public-key-format"';
        }
        refine "local-or-keystore/keystore/"
          + "keystore-reference" {
          must 'deref(..)/../ks:public-key-format'
            + ' = "ct:ssh-public-key-format"';
        }
      }
    }
  }
  container certificate {
    if-feature "sshcmn:ssh-x509-certs";
    description
      "A locally-defined or referenced end-entity
        certificate to be used for the SSH server's
        host key.";
    reference
      "RFC CCCC: A YANG Data Model for a Keystore";
    uses
      ks:local-or-keystore-end-entity-cert-with-key-grouping {
        refine "local-or-keystore/local/local-definition" {
          must 'public-key-format'
            + ' = "ct:subject-public-key-info-format"';
        }
        refine "local-or-keystore/keystore/keystore-reference"
          + "/asymmetric-key" {
          must 'deref(..)/../ks:public-key-format'
            + ' = "ct:subject-public-key-info-format"';
        }
      }
  }
```

```
    }
  }
}
} // container server-identity

container client-authentication {
  nacm:default-deny-write;
  description
    "Specifies how the SSH server can authenticate SSH clients.";
  container users {
    if-feature "local-users-supported";
    description
      "A list of locally configured users.";
    list user {
      key "name";
      description
        "A locally configured user.

        The server SHOULD derive the list of authentication
        'method names' returned to the SSH client from the
        descendant nodes configured herein, per Sections
        5.1 and 5.2 in RFC 4252.

        The authentication methods are unordered. Clients
        must authenticate to all configured methods.
        Whenever a choice amongst methods arises,
        implementations SHOULD use a default ordering
        that prioritizes automation over human-interaction.";
    leaf name {
      type string;
      description
        "The 'user name' for the SSH client, as defined in
        the SSH_MSG_USERAUTH_REQUEST message in RFC 4253.";
    }
  }
  container public-keys {
    if-feature "local-user-auth-publickey";
    presence
      "Indicates that public keys have been configured.
      This statement is present so the mandatory descendant
      nodes do not imply that this node must be
      configured.";
    description
      "A set of SSH public keys may be used by the SSH
      server to authenticate this user. A user is
      authenticated if its public key is an exact
      match to a configured public key.";
    reference
```



```

    "RFC BBBB: A YANG Data Model for a Truststore";
  uses ts:local-or-truststore-public-keys-grouping {
    refine "local-or-truststore/local/local-definition"
      + "/public-key" {
        must 'public-key-format'
          + ' = "ct:ssh-public-key-format"';
      }
    refine "local-or-truststore/truststore/"
      + "truststore-reference" {
        must 'deref(..)/../ts:public-key-format'
          + ' = "ct:ssh-public-key-format"';
      }
  }
}
leaf password {
  if-feature "local-user-auth-password";
  type ianach:crypt-hash;
  description
    "The password for this user.";
}
container hostbased {
  if-feature "local-user-auth-hostbased";
  presence
    "Indicates that hostbased keys have been configured.
    This statement is present so the mandatory descendant
    nodes do not imply that this node must be
    configured.";
  description
    "A set of SSH host keys used by the SSH server to
    authenticate this user's host. A user's host is
    authenticated if its host key is an exact match
    to a configured host key.";
  reference
    "RFC 4253: The Secure Shell (SSH) Transport Layer
    RFC BBBB: A YANG Data Model for a Truststore";
  uses ts:local-or-truststore-public-keys-grouping {
    refine "local-or-truststore/local/local-definition"
      + "/public-key" {
        must 'public-key-format'
          + ' = "ct:ssh-public-key-format"';
      }
    refine "local-or-truststore/truststore"
      + "/truststore-reference" {
        must 'deref(..)/../ts:public-key-format'
          + ' = "ct:ssh-public-key-format"';
      }
  }
}
}

```

```
    leaf none {
      if-feature "local-user-auth-none";
      type empty;
      description
        "Indicates that the 'none' method is configured
         for this user.";
      reference
        "RFC 4252: The Secure Shell (SSH) Authentication
         Protocol.";
    }
  }
}
container ca-certs {
  if-feature "sshcmn:ssh-x509-certs";
  presence
    "Indicates that CA certificates have been configured.
     This statement is present so the mandatory descendant
     nodes do not imply this node must be configured.";
  description
    "A set of certificate authority (CA) certificates used by
     the SSH server to authenticate SSH client certificates.
     A client certificate is authenticated if it has a valid
     chain of trust to a configured CA certificate.";
  reference
    "RFC BBBB: A YANG Data Model for a Truststore";
  uses ts:local-or-truststore-certs-grouping;
}
container ee-certs {
  if-feature "sshcmn:ssh-x509-certs";
  presence
    "Indicates that EE certificates have been configured.
     This statement is present so the mandatory descendant
     nodes do not imply this node must be configured.";
  description
    "A set of client certificates (i.e., end entity
     certificates) used by the SSH server to authenticate
     the certificates presented by SSH clients. A client
     certificate is authenticated if it is an exact match
     to a configured end-entity certificate.";
  reference
    "RFC BBBB: A YANG Data Model for a Truststore";
  uses ts:local-or-truststore-certs-grouping;
}
} // container client-authentication

container transport-params {
  nacm:default-deny-write;
  if-feature "sshcmn:transport-params";
```

```
    description
        "Configurable parameters of the SSH transport layer.";
    uses sshcmn:transport-params-grouping;
} // container transport-params

container keepalives {
    nacm:default-deny-write;
    if-feature "ssh-server-keepalives";
    presence
        "Indicates that the SSH server proactively tests the
         aliveness of the remote SSH client.";
    description
        "Configures the keep-alive policy, to proactively test
         the aliveness of the SSL client.  An unresponsive SSL
         client is dropped after approximately max-wait *
         max-attempts seconds.  Per Section 4 of RFC 4254,
         the SSH server SHOULD send an SSH_MSG_GLOBAL_REQUEST
         message with a purposely nonexistent 'request name'
         value (e.g., keepalive@ietf.org) and the 'want reply'
         value set to '1'.";
    reference
        "RFC 4254: The Secure Shell (SSH) Connection Protocol";
    leaf max-wait {
        type uint16 {
            range "1..max";
        }
        units "seconds";
        default "30";
        description
            "Sets the amount of time in seconds after which
             if no data has been received from the SSL client,
             a SSL-level message will be sent to test the
             aliveness of the SSL client.";
    }
    leaf max-attempts {
        type uint8;
        default "3";
        description
            "Sets the maximum number of sequential keep-alive
             messages that can fail to obtain a response from
             the SSL client before assuming the SSL client is
             no longer alive.";
    }
}
} // grouping ssh-server-grouping
}
```

<CODE ENDS>

## 5. Security Considerations

### 5.1. The "iana-ssh-key-exchange-algs" Module

The "iana-ssh-key-exchange-algs" YANG module defines a data model that 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.

This YANG module defines YANG identities, for a public IANA-maintained registry, and a single protocol-accessible read-only node for the subset of those identities supported by a server.

YANG identities are not security-sensitive, as they are statically defined in the publicly-accessible YANG module.

The protocol-accessible read-only node for the algorithms supported by a server is mildly sensitive, but not to the extent that special NACM annotations are needed to prevent read-access to regular authenticated administrators.

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

### 5.2. The "iana-ssh-encryption-algs" Module

The "iana-ssh-encryption-algs" YANG module defines a data model that 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.

This YANG module defines YANG identities, for a public IANA-maintained registry, and a single protocol-accessible read-only node for the subset of those identities supported by a server.

YANG identities are not security-sensitive, as they are statically defined in the publicly-accessible YANG module.

The protocol-accessible read-only node for the algorithms supported by a server is mildly sensitive, but not to the extent that special NACM annotations are needed to prevent read-access to regular authenticated administrators.

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

### 5.3. The "iana-ssh-mac-algs" Module

The "iana-ssh-mac-algs" YANG module defines a data model that 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.

This YANG module defines YANG identities, for a public IANA-maintained registry, and a single protocol-accessible read-only node for the subset of those identities supported by a server.

YANG identities are not security-sensitive, as they are statically defined in the publicly-accessible YANG module.

The protocol-accessible read-only node for the algorithms supported by a server is mildly sensitive, but not to the extent that special NACM annotations are needed to prevent read-access to regular authenticated administrators.

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

### 5.4. The "iana-ssh-public-key-algs" Module

The "iana-ssh-public-key-algs" YANG module defines a data model that 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.

This YANG module defines YANG identities, for a public IANA-maintained registry, and a single protocol-accessible read-only node for the subset of those identities supported by a server.

YANG identities are not security-sensitive, as they are statically defined in the publicly-accessible YANG module.

The protocol-accessible read-only node for the algorithms supported by a server is mildly sensitive, but not to the extent that special NACM annotations are needed to prevent read-access to regular authenticated administrators.

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

#### 5.5. The "ietf-ssh-common" YANG Module

The "ietf-ssh-common" YANG module defines "grouping" statements that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

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

Since this module only define groupings, these considerations are primarily for the designers of other modules that use these groupings.

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

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

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

## 5.6. The "ietf-ssh-client" YANG Module

The "ietf-ssh-client" YANG module defines "grouping" statements that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

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

Since this module only define groupings, these considerations are primarily for the designers of other modules that use these groupings.

One readable data node defined in this YANG module may be considered sensitive or vulnerable in some network environments. This node is as follows:

\* The "client-identity/password" node:

The cleartext "password" node defined in the "ssh-client-grouping" grouping is additionally sensitive to read operations such that, in normal use cases, it should never be returned to a client. For this reason, the NACM extension "default-deny-all" has been applied to it.

Please be aware that this module uses the "key" and "private-key" nodes from the "ietf-crypto-types" module [I-D.ietf-netconf-crypto-types], where said nodes have the NACM extension "default-deny-all" set, thus preventing unrestricted read-access to the cleartext key values.

All the writable data nodes defined by this module may be considered sensitive or vulnerable in some network environments. For instance, any modification to a key or reference to a key may dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been set for all data nodes defined in this module.

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

### 5.7. The "ietf-ssh-server" YANG Module

The "ietf-ssh-server" YANG module defines "grouping" statements that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

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

Since this module only define groupings, these considerations are primarily for the designers of other modules that use these groupings.

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

Please be aware that this module uses the "key" and "private-key" nodes from the "ietf-crypto-types" module [I-D.ietf-netconf-crypto-types], where said nodes have the NACM extension "default-deny-all" set, thus preventing unrestricted read-access to the cleartext key values.

All the writable data nodes defined by this module may be considered sensitive or vulnerable in some network environments. For instance, the addition or removal of references to keys, certificates, trusted anchors, etc., or even the modification of transport or keepalive parameters can dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been set for all data nodes defined in this module.

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

## 6. IANA Considerations

### 6.1. The "IETF XML" Registry

This document registers seven 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:iana-ssh-key-exchange-algs

Registrant Contact: IANA

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

URI: urn:ietf:params:xml:ns:yang:iana-ssh-encryption-algs

Registrant Contact: IANA

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

URI: urn:ietf:params:xml:ns:yang:iana-ssh-mac-algs

Registrant Contact: IANA

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

URI: urn:ietf:params:xml:ns:yang:iana-ssh-public-key-algs

Registrant Contact: IANA

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

URI: urn:ietf:params:xml:ns:yang:ietf-ssh-common

Registrant Contact: The IESG

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

URI: urn:ietf:params:xml:ns:yang:ietf-ssh-client

Registrant Contact: The IESG

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

URI: urn:ietf:params:xml:ns:yang:ietf-ssh-server

Registrant Contact: The IESG

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

## 6.2. The "YANG Module Names" Registry

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

name: iana-ssh-key-exchange-algs  
namespace: urn:ietf:params:xml:ns:yang:iana-ssh-key-exchange-algs  
prefix: sshkea  
reference: RFC EEEE

name: iana-ssh-encryption-algs  
namespace: urn:ietf:params:xml:ns:yang:iana-ssh-encryption-algs  
prefix: sshea  
reference: RFC EEEE

name: iana-ssh-mac-algs  
namespace: urn:ietf:params:xml:ns:yang:iana-ssh-mac-algs  
prefix: sshma  
reference: RFC EEEE

name: iana-ssh-public-key-algs  
namespace: urn:ietf:params:xml:ns:yang:iana-ssh-public-key-algs  
prefix: sshpka  
reference: RFC EEEE

name: ietf-ssh-common  
namespace: urn:ietf:params:xml:ns:yang:ietf-ssh-common  
prefix: sshcmn  
reference: RFC EEEE

name: ietf-ssh-client  
namespace: urn:ietf:params:xml:ns:yang:ietf-ssh-client  
prefix: sshc  
reference: RFC EEEE

name: ietf-ssh-server  
namespace: urn:ietf:params:xml:ns:yang:ietf-ssh-server  
prefix: sshs  
reference: RFC EEEE

### 6.3. The "iana-ssh-encryption-algs" Module

IANA is requested to maintain a YANG module called "iana-ssh-encryption-algs" that shadows the "Encryption Algorithm Names" sub-registry of the "Secure Shell (SSH) Protocol Parameters" registry [IANA-ENC-ALGS].

This registry defines a YANG identity for each encryption algorithm, and a "base" identity from which all of the other identities are derived.

An initial version of this module can be found in Appendix A.1

- \* Please note that this module was created on June 1st, 2021, and that additional entries may have been added in the interim before this document's publication. If this is that case, IANA may either publish just an updated module containing the new entries, or publish the initial module as is immediately followed by a "revision" containing the additional algorithm names.

#### 6.4. The "iana-ssh-mac-algs" Module

IANA is requested to maintain a YANG module called "iana-ssh-mac-algs" that shadows the "MAC Algorithm Names" sub-registry of the "Secure Shell (SSH) Protocol Parameters" registry [IANA-MAC-ALGS].

This registry defines a YANG identity for each MAC algorithm, and a "base" identity from which all of the other identities are derived.

An initial version of this module can be found in Appendix A.2.

- \* Please note that this module was created on June 1st, 2021, and that additional entries may have been added in the interim before this document's publication. If this is that case, IANA may either publish just an updated module containing the new entries, or publish the initial module as is immediately followed by a "revision" containing the additional algorithm names.

#### 6.5. The "iana-ssh-public-key-algs" Module

IANA is requested to maintain a YANG module called "iana-ssh-public-key-algs" that shadows the "Public Key Algorithm Names" sub-registry of the "Secure Shell (SSH) Protocol Parameters" registry [IANA-PUBKEY-ALGS].

This registry defines a YANG identity for each public key algorithm, and a "base" identity from which all of the other identities are derived.

Registry entries for which the '\*All values beginning with the specified string and not containing "@".' note applies MUST be expanded so that there is a distinct YANG identity for each enumeration.

An initial version of this module can be found in Appendix A.3.

- \* Please note that this module was created on June 1st, 2021, and that additional entries may have been added in the interim before this document's publication. If this is that case, IANA may either publish just an updated module containing the new entries, or publish the initial module as is immediately followed by a "revision" containing the additional algorithm names.

#### 6.6. The "iana-ssh-key-exchange-algs" Module

IANA is requested to maintain a YANG module called "iana-ssh-key-exchange-algs" that shadows the "Key Exchange Method Names" sub-registry of the "Secure Shell (SSH) Protocol Parameters" registry [IANA-KEYEX-ALGS].

This registry defines a YANG identity for each key exchange algorithm, and a "base" identity from which all of the other identities are derived.

Registry entries for which the '\*All values beginning with the specified string and not containing "@".' note applies MUST be expanded so that there is a distinct YANG identity for each enumeration.

An initial version of this module can be found in Appendix A.4.

- \* Please note that this module was created on June 1st, 2021, and that additional entries may have been added in the interim before this document's publication. If this is that case, IANA may either publish just an updated module containing the new entries, or publish the initial module as is immediately followed by a "revision" containing the additional algorithm names.
- \* Please also note that the "status" statement has been set to "deprecated" <https://datatracker.ietf.org/doc/html/rfc8732#section-6>. It is recommended that IANA adds a column to the registry to more easily track the deprecation status of algorithms.

### 7. References

#### 7.1. Normative References

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

Watson, K., "YANG Data Types and Groupings for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-21, 14 September 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-crypto-types-21>>.

[I-D.ietf-netconf-keystore]

Watsen, K., "A YANG Data Model for a Keystore", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-23, 14 December 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-keystore-23>>.

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

Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-16, 14 December 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-trust-anchors-16>>.

[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>>.

## 7.2. Informative References

- [I-D.ietf-netconf-http-client-server]  
Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-08, 14 December 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-http-client-server-08>>.
- [I-D.ietf-netconf-netconf-client-server]  
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-24, 14 December 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-netconf-client-server-24>>.
- [I-D.ietf-netconf-restconf-client-server]  
Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-24, 14 December 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-restconf-client-server-24>>.
- [I-D.ietf-netconf-ssh-client-server]  
Watsen, K., "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-ssh-client-server-26, 14 December 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-ssh-client-server-26>>.

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

Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tcp-client-server-11, 14 December 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tcp-client-server-11>>.

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

Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-26, 14 December 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tls-client-server-26>>.

[IANA-ENC-ALGS]

(IANA), I. A. N. A., "IANA "Encryption Algorithm Names" Sub-registry of the "Secure Shell (SSH) Protocol Parameters" Registry", <<https://www.iana.org/assignments/ssh-parameters/ssh-parameters.xhtml#ssh-parameters-17>>.

[IANA-KEYEX-ALGS]

(IANA), I. A. N. A., "IANA "Key Exchange Method Names" Sub-registry of the "Secure Shell (SSH) Protocol Parameters" Registry", <<https://www.iana.org/assignments/ssh-parameters/ssh-parameters.xhtml#ssh-parameters-16>>.

[IANA-MAC-ALGS]

(IANA), I. A. N. A., "IANA "MAC Algorithm Names" Sub-registry of the "Secure Shell (SSH) Protocol Parameters" Registry", <<https://www.iana.org/assignments/ssh-parameters/ssh-parameters.xhtml#ssh-parameters-18>>.

[IANA-PUBKEY-ALGS]

(IANA), I. A. N. A., "IANA "Public Key Algorithm Names" Sub-registry of the "Secure Shell (SSH) Protocol Parameters" Registry", <<https://www.iana.org/assignments/ssh-parameters/ssh-parameters.xhtml#ssh-parameters-19>>.

[OPENSSH] Project, T. O., "OpenSSH", <<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>>.
- [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. YANG Modules for IANA

The modules contained in this section were generated by scripts using the contents of the associated sub-registry as they existed on June 1st, 2021.

##### A.1. Initial Module for the "Encryption Algorithm Names" Registry



## A.1.1. Data Model Overview

This section provides an overview of the "iana-ssh-encryption-algs" module in terms of its identities and protocol-accessible nodes.

## A.1.1.1. Identities

The following diagram lists the base "identity" statements defined in the module, of which there is just one, and illustrates that all the derived identity statements are generated from the associated IANA-maintained registry [IANA-ENC-ALGS].

Identities:

```
+-- encryption-alg-base
   +-- <identity-name from IANA registry>
```

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

## A.1.1.2. Typedefs

The following diagram illustrates the "typedef" statements defined in the "iana-ssh-encryption-algs" module:

Typedefs:

```
identityref
+-- encryption-algorithm-ref
```

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

Comments:

- \* The typedef defined in the "iana-ssh-encryption-algs" module extends the "identityref" type defined in [RFC7950].

## A.1.1.3. Protocol-accessible Nodes

The following tree diagram [RFC8340] lists all the protocol-accessible nodes defined in the "iana-ssh-encryption-algs" module:

```
module: iana-ssh-encryption-algs
+--ro supported-algorithms
   +--ro supported-algorithm*   encryption-algorithm-ref
```

Comments:

- \* Protocol-accessible nodes are those nodes that are accessible when the module is "implemented", as described in Section 5.6.5 of [RFC7950].

#### A.1.2. Example Usage

The following example illustrates operational state data indicating the SSH encryption algorithms supported by the server:

```
<supported-algorithms
  xmlns="urn:ietf:params:xml:ns:yang:iana-ssh-encryption-algs"
  xmlns:sshea="urn:ietf:params:xml:ns:yang:iana-ssh-encryption-algs">
  <supported-algorithm>sshea:aes256-ctr</supported-algorithm>
  <supported-algorithm>sshea:aes256-cbc</supported-algorithm>
  <supported-algorithm>sshea:twofish256-cbc</supported-algorithm>
  <supported-algorithm>sshea:serpent256-cbc</supported-algorithm>
  <supported-algorithm>sshea:arcfour256</supported-algorithm>
  <supported-algorithm>sshea:serpent256-ctr</supported-algorithm>
  <supported-algorithm>sshea:aead-aes-256-gcm</supported-algorithm>
</supported-algorithms>
```

#### A.1.3. YANG Module

Following are the complete contents to the initial IANA-maintained YANG module. Please note that the date "2021-06-01" reflects the day on which the extraction occurred.

```
<CODE BEGINS> file "iana-ssh-encryption-algs@2021-06-01.yang"

module iana-ssh-encryption-algs {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:iana-ssh-encryption-algs";
  prefix sshea;

  organization
    "Internet Assigned Numbers Authority (IANA)";

  contact
    "Postal: ICANN
      12025 Waterfront Drive, Suite 300
      Los Angeles, CA 90094-2536
      United States of America
    Tel: +1 310 301 5800
    Email: iana@iana.org";

  description
    "This module defines identities for the encryption algorithms
    defined in the 'Encryption Algorithm Names' sub-registry of the
```

'Secure Shell (SSH) Protocol Parameters' registry maintained by IANA.

Copyright (c) 2021 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

The initial version of this YANG module is part of RFC EEEE (<https://www.rfc-editor.org/info/rfcEEEE>); see the RFC itself for full legal notices.";

```
revision 2021-06-01 {
  description
    "Initial version";
  reference
    "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
}

// Typedefs

typedef encryption-algorithm-ref {
  type identityref {
    base "encryption-alg-base";
  }
  description
    "A reference to a SSH encryption algorithm identifier.";
}

// Identities

identity encryption-alg-base {
  description
    "Base identity used to identify encryption algorithms.";
}

identity triple-des-cbc { // YANG IDs cannot begin with a number
  base encryption-alg-base;
  description
    "3DES-CBC";
  reference
    "RFC 4253:"
```

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

    identity blowfish-cbc {
        base encryption-alg-base;
        description
            "BLOWFISH-CBC";
        reference
            "RFC 4253:
                The Secure Shell (SSH) Transport Layer Protocol";
    }

    identity twofish256-cbc {
        base encryption-alg-base;
        description
            "TWOFISH256-CBC";
        reference
            "RFC 4253:
                The Secure Shell (SSH) Transport Layer Protocol";
    }

    identity twofish-cbc {
        base encryption-alg-base;
        description
            "TWOFISH-CBC";
        reference
            "RFC 4253:
                The Secure Shell (SSH) Transport Layer Protocol";
    }

    identity twofish192-cbc {
        base encryption-alg-base;
        description
            "TWOFISH192-CBC";
        reference
            "RFC 4253:
                The Secure Shell (SSH) Transport Layer Protocol";
    }

    identity twofish128-cbc {
        base encryption-alg-base;
        description
            "TWOFISH128-CBC";
        reference
            "RFC 4253:
                The Secure Shell (SSH) Transport Layer Protocol";
    }
```

```
identity aes256-cbc {
  base encryption-alg-base;
  description
    "AES256-CBC";
  reference
    "RFC 4253:
      The Secure Shell (SSH) Transport Layer Protocol";
}

identity aes192-cbc {
  base encryption-alg-base;
  description
    "AES192-CBC";
  reference
    "RFC 4253:
      The Secure Shell (SSH) Transport Layer Protocol";
}

identity aes128-cbc {
  base encryption-alg-base;
  description
    "AES128-CBC";
  reference
    "RFC 4253:
      The Secure Shell (SSH) Transport Layer Protocol";
}

identity serpent256-cbc {
  base encryption-alg-base;
  description
    "SERPENT256-CBC";
  reference
    "RFC 4253:
      The Secure Shell (SSH) Transport Layer Protocol";
}

identity serpent192-cbc {
  base encryption-alg-base;
  description
    "SERPENT192-CBC";
  reference
    "RFC 4253:
      The Secure Shell (SSH) Transport Layer Protocol";
}

identity serpent128-cbc {
  base encryption-alg-base;
  description
```

```
    "SERPENT128-CBC";
  reference
    "RFC 4253:
      The Secure Shell (SSH) Transport Layer Protocol";
}

identity arcfour {
  base encryption-alg-base;
  status obsolete;
  description
    "ARCFOUR";
  reference
    "RFC 8758:
      Deprecating RC4 in Secure Shell (SSH)";
}

identity idea-cbc {
  base encryption-alg-base;
  description
    "IDEA-CBC";
  reference
    "RFC 4253:
      The Secure Shell (SSH) Transport Layer Protocol";
}

identity cast128-cbc {
  base encryption-alg-base;
  description
    "CAST128-CBC";
  reference
    "RFC 4253:
      The Secure Shell (SSH) Transport Layer Protocol";
}

identity none {
  base encryption-alg-base;
  description
    "NONE";
  reference
    "RFC 4253:
      The Secure Shell (SSH) Transport Layer Protocol";
}

identity des-cbc {
  base encryption-alg-base;
  status obsolete;
  description
    "DES-CBC";
```

```
    reference
      "FIPS 46-3:
        Data Encryption Standard (DES)";
  }

  identity arcfour128 {
    base encryption-alg-base;
    status obsolete;
    description
      "ARCFOUR128";
    reference
      "RFC 8758:
        Deprecating RC4 in Secure Shell (SSH)";
  }

  identity arcfour256 {
    base encryption-alg-base;
    status obsolete;
    description
      "ARCFOUR256";
    reference
      "RFC 8758:
        Deprecating RC4 in Secure Shell (SSH)";
  }

  identity aes128-ctr {
    base encryption-alg-base;
    description
      "AES128-CTR";
    reference
      "RFC 4344:
        The Secure Shell (SSH) Transport Layer Encryption Modes";
  }

  identity aes192-ctr {
    base encryption-alg-base;
    description
      "AES192-CTR";
    reference
      "RFC 4344:
        The Secure Shell (SSH) Transport Layer Encryption Modes";
  }

  identity aes256-ctr {
    base encryption-alg-base;
    description
      "AES256-CTR";
    reference
```

```
    "RFC 4344:
      The Secure Shell (SSH) Transport Layer Encryption Modes";
  }

  identity triple-des-ctr { // YANG IDs cannot begin with a number
    base encryption-alg-base;
    description
      "3DES-CTR";
    reference
      "RFC 4344:
        The Secure Shell (SSH) Transport Layer Encryption Modes";
  }

  identity blowfish-ctr {
    base encryption-alg-base;
    description
      "BLOWFISH-CTR";
    reference
      "RFC 4344:
        The Secure Shell (SSH) Transport Layer Encryption Modes";
  }

  identity twofish128-ctr {
    base encryption-alg-base;
    description
      "TWOFISH128-CTR";
    reference
      "RFC 4344:
        The Secure Shell (SSH) Transport Layer Encryption Modes";
  }

  identity twofish192-ctr {
    base encryption-alg-base;
    description
      "TWOFISH192-CTR";
    reference
      "RFC 4344:
        The Secure Shell (SSH) Transport Layer Encryption Modes";
  }

  identity twofish256-ctr {
    base encryption-alg-base;
    description
      "TWOFISH256-CTR";
    reference
      "RFC 4344:
        The Secure Shell (SSH) Transport Layer Encryption Modes";
  }
}
```



```
identity serpent128-ctr {
  base encryption-alg-base;
  description
    "SERPENT128-CTR";
  reference
    "RFC 4344:
      The Secure Shell (SSH) Transport Layer Encryption Modes";
}

identity serpent192-ctr {
  base encryption-alg-base;
  description
    "SERPENT192-CTR";
  reference
    "RFC 4344:
      The Secure Shell (SSH) Transport Layer Encryption Modes";
}

identity serpent256-ctr {
  base encryption-alg-base;
  description
    "SERPENT256-CTR";
  reference
    "RFC 4344:
      The Secure Shell (SSH) Transport Layer Encryption Modes";
}

identity idea-ctr {
  base encryption-alg-base;
  description
    "IDEA-CTR";
  reference
    "RFC 4344:
      The Secure Shell (SSH) Transport Layer Encryption Modes";
}

identity cast128-ctr {
  base encryption-alg-base;
  description
    "CAST128-CTR";
  reference
    "RFC 4344:
      The Secure Shell (SSH) Transport Layer Encryption Modes";
}

identity aead-aes-128-gcm {
  base encryption-alg-base;
  description
```

```
        "AEAD_AES_128_GCM";
    reference
        "RFC 5647:
        AES Galois Counter Mode for the
        Secure Shell Transport Layer Protocol";
}

identity aead-aes-256-gcm {
    base encryption-alg-base;
    description
        "AEAD_AES_256_GCM";
    reference
        "RFC 5647:
        AES Galois Counter Mode for the
        Secure Shell Transport Layer Protocol";
}

// Protocol-accessible Nodes

container supported-algorithms {
    config false;
    description
        "A container for a list of encryption algorithms
        supported by the server.";
    leaf-list supported-algorithm {
        type encryption-algorithm-ref;
        description
            "A encryption algorithm supported by the server.";
    }
}

}

<CODE ENDS>
```

## A.2. Initial Module for the "MAC Algorithm Names" Registry

### A.2.1. Data Model Overview

This section provides an overview of the "iana-ssh-mac-algs" module in terms of its identities and protocol-accessible nodes.

#### A.2.1.1. Identities

The following diagram lists the base "identity" statements defined in the module, of which there is just one, and illustrates that all the derived identity statements are generated from the associated IANA-maintained registry [IANA-MAC-ALGS].

## Identities:

```
+-- mac-alg-base
   +-- <identity-name from IANA registry>
```

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

## A.2.1.2. Typedefs

The following diagram illustrates the "typedef" statements defined in the "iana-ssh-mac-algs" module:

## Typedefs:

```
identityref
   +-- mac-algorithm-ref
```

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

## Comments:

- \* The typedef defined in the "iana-ssh-mac-algs" module extends the "identityref" type defined in [RFC7950].

## A.2.1.3. Protocol-accessible Nodes

The following tree diagram [RFC8340] lists all the protocol-accessible nodes defined in the "iana-ssh-mac-algs" module:

```
module: iana-ssh-mac-algs
   +--ro supported-algorithms
      +--ro supported-algorithm*   mac-algorithm-ref
```

## Comments:

- \* Protocol-accessible nodes are those nodes that are accessible when the module is "implemented", as described in Section 5.6.5 of [RFC7950].

## A.2.2. Example Usage

The following example illustrates operational state data indicating the SSH MAC algorithms supported by the server:

```
<supported-algorithms
  xmlns="urn:ietf:params:xml:ns:yang:iana-ssh-mac-algs"
  xmlns:sshma="urn:ietf:params:xml:ns:yang:iana-ssh-mac-algs">
  <supported-algorithm>sshma:hmac-sha2-256</supported-algorithm>
  <supported-algorithm>sshma:hmac-sha2-512</supported-algorithm>
  <supported-algorithm>sshma:aead-aes-256-gcm</supported-algorithm>
</supported-algorithms>
```

#### A.2.3. YANG Module

Following are the complete contents to the initial IANA-maintained YANG module. Please note that the date "2021-06-01" reflects the day on which the extraction occurred.

```
<CODE BEGINS> file "iana-ssh-mac-algs@2021-06-01.yang"
```

```
module iana-ssh-mac-algs {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:iana-ssh-mac-algs";
  prefix sshma;

  organization
    "Internet Assigned Numbers Authority (IANA)";

  contact
    "Postal: ICANN
      12025 Waterfront Drive, Suite 300
      Los Angeles, CA 90094-2536
      United States of America
    Tel: +1 310 301 5800
    Email: iana@iana.org";

  description
    "This module defines identities for the MAC algorithms
    defined in the 'MAC Algorithm Names' sub-registry of the
    'Secure Shell (SSH) Protocol Parameters' registry maintained
    by IANA.

    Copyright (c) 2021 IETF Trust and the persons identified as
    authors of the code. All rights reserved.

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

The initial version of this YANG module is part of RFC EEEE (<https://www.rfc-editor.org/info/rfcEEEE>); see the RFC itself for full legal notices.";

```
revision 2021-06-01 {
  description
    "Initial version";
  reference
    "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
}

// Typedefs

typedef mac-algorithm-ref {
  type identityref {
    base "mac-alg-base";
  }
  description
    "A reference to a SSH mac algorithm identifier.";
}

// Identities

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-sha1-96 {
  base mac-alg-base;
  description
    "HMAC-SHA1-96";
  reference
    "RFC 4253:
    The Secure Shell (SSH) Transport Layer Protocol";
}
```

```
identity hmac-md5 {
  base mac-alg-base;
  description
    "HMAC-MD5";
  reference
    "RFC 4253:
      The Secure Shell (SSH) Transport Layer Protocol";
}

identity hmac-md5-96 {
  base mac-alg-base;
  description
    "HMAC-MD5-96";
  reference
    "RFC 4253:
      The Secure Shell (SSH) Transport Layer Protocol";
}

identity none {
  base mac-alg-base;
  description
    "NONE";
  reference
    "RFC 4253:
      The Secure Shell (SSH) Transport Layer Protocol";
}

identity aead-aes-128-gcm {
  base mac-alg-base;
  description
    "AEAD_AES_128_GCM";
  reference
    "RFC 5647:
      AES Galois Counter Mode for the
      Secure Shell Transport Layer Protocol";
}

identity aead-aes-256-gcm {
  base mac-alg-base;
  description
    "AEAD_AES_256_GCM";
  reference
    "RFC 5647:
      AES Galois Counter Mode for the
      Secure Shell Transport Layer Protocol";
}

identity hmac-sha2-256 {
```

```
    base mac-alg-base;
    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;
    description
        "HMAC-SHA2-512";
    reference
        "RFC 6668:
        SHA-2 Data Integrity Verification for the
        Secure Shell (SSH) Transport Layer Protocol";
}

// Protocol-accessible Nodes

container supported-algorithms {
    config false;
    description
        "A container for a list of MAC algorithms
        supported by the server.";
    leaf-list supported-algorithm {
        type mac-algorithm-ref;
        description
            "A MAC algorithm supported by the server.";
    }
}

}

<CODE ENDS>
```

### A.3. Initial Module for the "Public Key Algorithm Names" Registry

#### A.3.1. Data Model Overview

This section provides an overview of the "iana-ssh-public-key-algs" module in terms of its identities and protocol-accessible nodes.

## A.3.1.1. Identities

The following diagram lists the base "identity" statements defined in the module, of which there is just one, and illustrates that all the derived identity statements are generated from the associated IANA-maintained registry [IANA-PUBKEY-ALGS].

Identities:

```
+-- public-key-alg-base
   +-- <identity-name from IANA registry>
```

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

## A.3.1.2. Typedefs

The following diagram illustrates the "typedef" statements defined in the "iana-ssh-public-key-algs" module:

Typedefs:

```
identityref
   +-- public-key-algorithm-ref
```

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

Comments:

- \* The typedef defined in the "iana-ssh-public-key-algs" module extends the "identityref" type defined in [RFC7950].

## A.3.1.3. Protocol-accessible Nodes

The following tree diagram [RFC8340] lists all the protocol-accessible nodes defined in the "iana-ssh-public-key-algs" module:

```
module: iana-ssh-public-key-algs
   +--ro supported-algorithms
      +--ro supported-algorithm*   public-key-algorithm-ref
```

Comments:

- \* Protocol-accessible nodes are those nodes that are accessible when the module is "implemented", as described in Section 5.6.5 of [RFC7950].



### A.3.2. Example Usage

The following example illustrates operational state data indicating the SSH public key algorithms supported by the server:

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

```
<supported-algorithms
  xmlns="urn:ietf:params:xml:ns:yang:iana-ssh-public-key-algs"
  xmlns:sshpk="urn:ietf:params:xml:ns:yang:iana-ssh-public-key-algs\
">
  <supported-algorithm>sshpk:rsa-sha2-256</supported-algorithm>
  <supported-algorithm>sshpk:rsa-sha2-512</supported-algorithm>
  <supported-algorithm>sshpk:spki-sign-rsa</supported-algorithm>
  <supported-algorithm>sshpk:pgp-sign-dss</supported-algorithm>
  <supported-algorithm>sshpk:x509v3-rsa2048-sha256</supported-algor\
ithm>
  <supported-algorithm>sshpk:ecdsa-sha2-nistp256</supported-algorit\
hm>
  <supported-algorithm>sshpk:ecdsa-sha2-1.3.132.0.37</supported-alg\
orithm>
  <supported-algorithm>sshpk:ssh-ed25519</supported-algorithm>
</supported-algorithms>
```

### A.3.3. YANG Module

Following are the complete contents to the initial IANA-maintained YANG module. Please note that the date "2021-06-01" reflects the day on which the extraction occurred.

```
<CODE BEGINS> file "iana-ssh-public-key-algs@2021-06-01.yang"

module iana-ssh-public-key-algs {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:iana-ssh-public-key-algs";
  prefix sshpk;

  organization
    "Internet Assigned Numbers Authority (IANA)";

  contact
    "Postal: ICANN
      12025 Waterfront Drive, Suite 300
      Los Angeles, CA 90094-2536
      United States of America
      Tel: +1 310 301 5800
      Email: iana@iana.org";
```

## description

"This module defines identities for the public key algorithms defined in the 'Public Key Algorithm Names' sub-registry of the 'Secure Shell (SSH) Protocol Parameters' registry maintained by IANA.

Copyright (c) 2021 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

The initial version of this YANG module is part of RFC EEEE (<https://www.rfc-editor.org/info/rfcEEEE>); see the RFC itself for full legal notices.";

```
revision 2021-06-01 {  
  description  
    "Initial version";  
  reference  
    "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";  
}
```

```
// Typedefs
```

```
typedef public-key-algorithm-ref {  
  type identityref {  
    base "public-key-alg-base";  
  }  
  description  
    "A reference to a SSH public key algorithm identifier.";  
}
```

```
// Identities
```

```
identity public-key-alg-base {  
  description  
    "Base identity used to identify public key algorithms.";  
}
```

```
identity ssh-dss {  
  base public-key-alg-base;  
  description
```

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

identity ssh-rsa {
    base public-key-alg-base;
    description
        "SSH-RSA";
    reference
        "RFC 4253:
        The Secure Shell (SSH) Transport Layer Protocol";
}

identity rsa-sha2-256 {
    base public-key-alg-base;
    description
        "RSA-SHA2-256";
    reference
        "RFC 8332:
        Use of RSA Keys with SHA-256 and SHA-512
        in the Secure Shell (SSH) Protocol";
}

identity rsa-sha2-512 {
    base public-key-alg-base;
    description
        "RSA-SHA2-512";
    reference
        "RFC 8332:
        Use of RSA Keys with SHA-256 and SHA-512
        in the Secure Shell (SSH) Protocol";
}

identity spki-sign-rsa {
    base public-key-alg-base;
    description
        "SPKI-SIGN-RSA";
    reference
        "RFC 4253:
        The Secure Shell (SSH) Transport Layer Protocol";
}

identity spki-sign-dss {
    base public-key-alg-base;
    description
        "SPKI-SIGN-DSS";
```

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

  identity pgp-sign-rsa {
    base public-key-alg-base;
    description
      "PGP-SIGN-RSA";
    reference
      "RFC 4253:
        The Secure Shell (SSH) Transport Layer Protocol";
  }

  identity pgp-sign-dss {
    base public-key-alg-base;
    description
      "PGP-SIGN-DSS";
    reference
      "RFC 4253:
        The Secure Shell (SSH) Transport Layer Protocol";
  }

  identity null {
    base public-key-alg-base;
    description
      "NULL";
    reference
      "RFC 4462:
        Generic Security Service Application Program Interface
        (GSS-API) Authentication and Key Exchange for the
        Secure Shell (SSH) Protocol";
  }

  identity ecdsa-sha2-nistp256 {
    base public-key-alg-base;
    description
      "ECDSA-SHA2-NISTP256 (secp256r1)";
    reference
      "RFC 5656:
        Elliptic Curve Algorithm Integration in the
        Secure Shell Transport Layer";
  }

  identity ecdsa-sha2-nistp384 {
    base public-key-alg-base;
    description
      "ECDSA-SHA2-NISTP384 (secp384r1)";
```

```
    reference
      "RFC 5656:
        Elliptic Curve Algorithm Integration in the
        Secure Shell Transport Layer";
  }

  identity ecdsa-sha2-nistp521 {
    base public-key-alg-base;
    description
      "ECDSA-SHA2-NISTP521 (secp521r1)";
    reference
      "RFC 5656:
        Elliptic Curve Algorithm Integration in the
        Secure Shell Transport Layer";
  }

  identity ecdsa-sha2-1.3.132.0.1 {
    base public-key-alg-base;
    description
      "ECDSA-SHA2-1.3.132.0.1 (nistk163, sect163k1)";
    reference
      "RFC 5656:
        Elliptic Curve Algorithm Integration in the
        Secure Shell Transport Layer";
  }

  identity ecdsa-sha2-1.2.840.10045.3.1.1 {
    base public-key-alg-base;
    description
      "ECDSA-SHA2-1.2.840.10045.3.1.1 (nistp192, secp192r1)";
    reference
      "RFC 5656:
        Elliptic Curve Algorithm Integration in the
        Secure Shell Transport Layer";
  }

  identity ecdsa-sha2-1.3.132.0.33 {
    base public-key-alg-base;
    description
      "ECDSA-SHA2-1.3.132.0.33 (nistp224, secp224r1)";
    reference
      "RFC 5656:
        Elliptic Curve Algorithm Integration in the
        Secure Shell Transport Layer";
  }

  identity ecdsa-sha2-1.3.132.0.26 {
    base public-key-alg-base;
```

```
    description
      "ECDSA-SHA2-1.3.132.0.26 (nistk233, sect233k1)";
    reference
      "RFC 5656:
        Elliptic Curve Algorithm Integration in the
        Secure Shell Transport Layer";
  }

  identity ecdsa-sha2-1.3.132.0.27 {
    base public-key-alg-base;
    description
      "ECDSA-SHA2-1.3.132.0.27 (nistb233, sect233r1)";
    reference
      "RFC 5656:
        Elliptic Curve Algorithm Integration in the
        Secure Shell Transport Layer";
  }

  identity ecdsa-sha2-1.3.132.0.16 {
    base public-key-alg-base;
    description
      "ECDSA-SHA2-1.3.132.0.16 (nistk283, sect283k1)";
    reference
      "RFC 5656:
        Elliptic Curve Algorithm Integration in the
        Secure Shell Transport Layer";
  }

  identity ecdsa-sha2-1.3.132.0.36 {
    base public-key-alg-base;
    description
      "ECDSA-SHA2-1.3.132.0.36 (nistk409, sect409k1)";
    reference
      "RFC 5656:
        Elliptic Curve Algorithm Integration in the
        Secure Shell Transport Layer";
  }

  identity ecdsa-sha2-1.3.132.0.37 {
    base public-key-alg-base;
    description
      "ECDSA-SHA2-1.3.132.0.37 (nistb409, sect409r1)";
    reference
      "RFC 5656:
        Elliptic Curve Algorithm Integration in the
        Secure Shell Transport Layer";
  }
```

```
identity ecdsa-sha2-1.3.132.0.38 {
  base public-key-alg-base;
  description
    "ECDSA-SHA2-1.3.132.0.38 (nistt571, sect571k1)";
  reference
    "RFC 5656:
      Elliptic Curve Algorithm Integration in the
      Secure Shell Transport Layer";
}

identity x509v3-ssh-dss {
  base public-key-alg-base;
  description
    "X509V3-SSH-DSS";
  reference
    "RFC 6187:
      X.509v3 Certificates for Secure Shell Authentication";
}

identity x509v3-ssh-rsa {
  base public-key-alg-base;
  description
    "X509V3-SSH-RSA";
  reference
    "RFC 6187:
      X.509v3 Certificates for Secure Shell Authentication";
}

identity x509v3-rsa2048-sha256 {
  base public-key-alg-base;
  description
    "X509V3-RSA2048-SHA256";
  reference
    "RFC 6187:
      X.509v3 Certificates for Secure Shell Authentication";
}

identity x509v3-ecdsa-sha2-nistp256 {
  base public-key-alg-base;
  description
    "X509V3-ECDSA-SHA2-NISTP256 (secp256r1)";
  reference
    "RFC 6187:
      X.509v3 Certificates for Secure Shell Authentication";
}

identity x509v3-ecdsa-sha2-nistp384 {
  base public-key-alg-base;
```

```
    description
      "X509V3-ECDSA-SHA2-NISTP384 (secp384r1)";
    reference
      "RFC 6187:
        X.509v3 Certificates for Secure Shell Authentication";
  }

  identity x509v3-ecdsa-sha2-nistp521 {
    base public-key-alg-base;
    description
      "X509V3-ECDSA-SHA2-NISTP521 (secp521r1)";
    reference
      "RFC 6187:
        X.509v3 Certificates for Secure Shell Authentication";
  }

  identity x509v3-ecdsa-sha2-1.3.132.0.1 {
    base public-key-alg-base;
    description
      "X509V3-ECDSA-SHA2-1.3.132.0.1 (nistk163, sect163k1)";
    reference
      "RFC 6187:
        X.509v3 Certificates for Secure Shell Authentication";
  }

  identity x509v3-ecdsa-sha2-1.2.840.10045.3.1.1 {
    base public-key-alg-base;
    description
      "X509V3-ECDSA-SHA2-1.2.840.10045.3.1.1 (nistp192, secp192r1)";
    reference
      "RFC 6187:
        X.509v3 Certificates for Secure Shell Authentication";
  }

  identity x509v3-ecdsa-sha2-1.3.132.0.33 {
    base public-key-alg-base;
    description
      "X509V3-ECDSA-SHA2-1.3.132.0.33 (nistp224, secp224r1)";
    reference
      "RFC 6187:
        X.509v3 Certificates for Secure Shell Authentication";
  }

  identity x509v3-ecdsa-sha2-1.3.132.0.26 {
    base public-key-alg-base;
    description
      "X509V3-ECDSA-SHA2-1.3.132.0.26 (nistk233, sect233k1)";
    reference
```



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

  identity x509v3-ecdsa-sha2-1.3.132.0.27 {
    base public-key-alg-base;
    description
      "X509V3-ECDSA-SHA2-1.3.132.0.27 (nistb233, sect233r1)";
    reference
      "RFC 6187:
        X.509v3 Certificates for Secure Shell Authentication";
  }

  identity x509v3-ecdsa-sha2-1.3.132.0.16 {
    base public-key-alg-base;
    description
      "X509V3-ECDSA-SHA2-1.3.132.0.16 (nistk283, sect283k1)";
    reference
      "RFC 6187:
        X.509v3 Certificates for Secure Shell Authentication";
  }

  identity x509v3-ecdsa-sha2-1.3.132.0.36 {
    base public-key-alg-base;
    description
      "X509V3-ECDSA-SHA2-1.3.132.0.36 (nistk409, sect409k1)";
    reference
      "RFC 6187:
        X.509v3 Certificates for Secure Shell Authentication";
  }

  identity x509v3-ecdsa-sha2-1.3.132.0.37 {
    base public-key-alg-base;
    description
      "X509V3-ECDSA-SHA2-1.3.132.0.37 (nistb409, sect409r1)";
    reference
      "RFC 6187:
        X.509v3 Certificates for Secure Shell Authentication";
  }

  identity x509v3-ecdsa-sha2-1.3.132.0.38 {
    base public-key-alg-base;
    description
      "X509V3-ECDSA-SHA2-1.3.132.0.38 (nistt571, sect571k1)";
    reference
      "RFC 6187:
        X.509v3 Certificates for Secure Shell Authentication";
  }
```

```
identity ssh-ed25519 {
  base public-key-alg-base;
  description
    "SSH-ED25519";
  reference
    "RFC 8709:
      Ed25519 and Ed448 Public Key Algorithms for the
      Secure Shell (SSH) Protocol";
}

identity ssh-ed448 {
  base public-key-alg-base;
  description
    "SSH-ED448";
  reference
    "RFC 8709:
      Ed25519 and Ed448 Public Key Algorithms for the
      Secure Shell (SSH) Protocol";
}

// Protocol-accessible Nodes

container supported-algorithms {
  config false;
  description
    "A container for a list of public key algorithms
    supported by the server.";
  leaf-list supported-algorithm {
    type public-key-algorithm-ref;
    description
      "A public key algorithm supported by the server.";
  }
}

}

<CODE ENDS>
```

#### A.4. Initial Module for the "Key Exchange Method Names" Registry

##### A.4.1. Data Model Overview

This section provides an overview of the "iana-ssh-key-exchange-algs" module in terms of its identities and protocol-accessible nodes.

## A.4.1.1. Identities

The following diagram lists the base "identity" statements defined in the module, of which there is just one, and illustrates that all the derived identity statements are generated from the associated IANA-maintained registry [IANA-KEYEX-ALGS].

Identities:

```
+-- key-exchange-alg-base
   +-- <identity-name from IANA registry>
```

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

## A.4.1.2. Typedefs

The following diagram illustrates the "typedef" statements defined in the "iana-ssh-key-exchange-algs" module:

Typedefs:

```
identityref
   +-- key-exchange-algorithm-ref
```

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

Comments:

- \* The typedef defined in the "iana-ssh-key-exchange-algs" module extends the "identityref" type defined in [RFC7950].

## A.4.1.3. Protocol-accessible Nodes

The following tree diagram [RFC8340] lists all the protocol-accessible nodes defined in the "iana-ssh-key-exchange-algs" module:

```
module: iana-ssh-key-exchange-algs
   +--ro supported-algorithms
      +--ro supported-algorithm*   key-exchange-algorithm-ref
```

Comments:

- \* Protocol-accessible nodes are those nodes that are accessible when the module is "implemented", as described in Section 5.6.5 of [RFC7950].

#### A.4.2. Example Usage

The following example illustrates operational state data indicating the SSH key exchange algorithms supported by the server:

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

```
<supported-algorithms
  xmlns="urn:ietf:params:xml:ns:yang:iana-ssh-key-exchange-algs"
  xmlns:sshkea="urn:ietf:params:xml:ns:yang:iana-ssh-key-exchange-al\
gs">
  <supported-algorithm>sshkea:diffie-hellman-group-exchange-sha256</\
supported-algorithm>
  <supported-algorithm>sshkea:ecdh-sha2-nistp256</supported-algorith\
m>
  <supported-algorithm>sshkea:rsa2048-sha256</supported-algorithm>
  <supported-algorithm>sshkea:gss-group1-sha1-curve25519-sha256</sup\
ported-algorithm>
  <supported-algorithm>sshkea:gss-group14-sha1-nistp256</supported-a\
lgorithm>
  <supported-algorithm>sshkea:gss-gex-sha1-nistp256</supported-algor\
ithm>
  <supported-algorithm>sshkea:gss-group14-sha256-1.2.840.10045.3.1.1\
</supported-algorithm>
  <supported-algorithm>sshkea:curve25519-sha256</supported-algorithm>
</supported-algorithms>
```

#### A.4.3. YANG Module

Following are the complete contents to the initial IANA-maintained YANG module. Please note that the date "2021-06-01" reflects the day on which the extraction occurred.

```
<CODE BEGINS> file "iana-ssh-key-exchange-algs@2021-06-01.yang"

module iana-ssh-key-exchange-algs {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:iana-ssh-key-exchange-algs";
  prefix sshkea;

  organization
    "Internet Assigned Numbers Authority (IANA)";

  contact
    "Postal: ICANN
      12025 Waterfront Drive, Suite 300
      Los Angeles, CA 90094-2536
      United States of America
```

Tel: +1 310 301 5800  
Email: iana@iana.org";

description

"This module defines identities for the key exchange algorithms defined in the 'Key Exchange Method Names' sub-registry of the 'Secure Shell (SSH) Protocol Parameters' registry maintained by IANA.

Copyright (c) 2021 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

The initial version of this YANG module is part of RFC EEEE (<https://www.rfc-editor.org/info/rfcEEEE>); see the RFC itself for full legal notices.";

```
revision 2021-06-01 {  
  description  
    "Initial version";  
  reference  
    "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";  
}
```

// Typedefs

```
typedef key-exchange-algorithm-ref {  
  type identityref {  
    base "key-exchange-alg-base";  
  }  
  description  
    "A reference to a SSH key exchange algorithm identifier.";  
}
```

// Identities

```
identity key-exchange-alg-base {  
  description  
    "Base identity used to identify key exchange algorithms.";  
}
```

```
identity diffie-hellman-group-exchange-sha1 {
  base key-exchange-alg-base;
  description
    "DIFFIE-HELLMAN-GROUP-EXCHANGE-SHA1";
  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;
  description
    "DIFFIE-HELLMAN-GROUP-EXCHANGE-SHA256";
  reference
    "RFC 4419:
      Diffie-Hellman Group Exchange for the
      Secure Shell (SSH) Transport Layer Protocol";
}

identity diffie-hellman-group1-sha1 {
  base key-exchange-alg-base;
  description
    "DIFFIE-HELLMAN-GROUP1-SHA1";
  reference
    "RFC 4253:
      The Secure Shell (SSH) Transport Layer Protocol";
}

identity diffie-hellman-group14-sha1 {
  base key-exchange-alg-base;
  description
    "DIFFIE-HELLMAN-GROUP14-SHA1";
  reference
    "RFC 4253:
      The Secure Shell (SSH) Transport Layer Protocol";
}

identity diffie-hellman-group14-sha256 {
  base key-exchange-alg-base;
  description
    "DIFFIE-HELLMAN-GROUP14-SHA256";
  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-alg-base;
  description
    "DIFFIE-HELLMAN-GROUP15-SHA512";
  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-alg-base;
  description
    "DIFFIE-HELLMAN-GROUP16-SHA512";
  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-alg-base;
  description
    "DIFFIE-HELLMAN-GROUP17-SHA512";
  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-alg-base;
  description
    "DIFFIE-HELLMAN-GROUP18-SHA512";
  reference
    "RFC 8268:
      More Modular Exponentiation (MODP) Diffie-Hellman (DH)
      Key Exchange (KEX) Groups for Secure Shell (SSH)";
}

identity ecdh-sha2-nistp256 {
  base key-exchange-alg-base;
  description
    "ECDH-SHA2-NISTP256 (secp256r1)";
  reference
    "RFC 5656:
      Elliptic Curve Algorithm Integration in the
      Secure Shell Transport Layer";
}
```

```
}

identity ecdh-sha2-nistp384 {
  base key-exchange-alg-base;
  description
    "ECDH-SHA2-NISTP384 (secp384r1)";
  reference
    "RFC 5656:
      Elliptic Curve Algorithm Integration in the
      Secure Shell Transport Layer";
}

identity ecdh-sha2-nistp521 {
  base key-exchange-alg-base;
  description
    "ECDH-SHA2-NISTP521 (secp521r1)";
  reference
    "RFC 5656:
      Elliptic Curve Algorithm Integration in the
      Secure Shell Transport Layer";
}

identity ecdh-sha2-1.3.132.0.1 {
  base key-exchange-alg-base;
  description
    "ECDH-SHA2-1.3.132.0.1 (nistk163, sect163k1)";
  reference
    "RFC 5656:
      Elliptic Curve Algorithm Integration in the
      Secure Shell Transport Layer";
}

identity ecdh-sha2-1.2.840.10045.3.1.1 {
  base key-exchange-alg-base;
  description
    "ECDH-SHA2-1.2.840.10045.3.1.1 (nistp192, secp192r1)";
  reference
    "RFC 5656:
      Elliptic Curve Algorithm Integration in the
      Secure Shell Transport Layer";
}

identity ecdh-sha2-1.3.132.0.33 {
  base key-exchange-alg-base;
  description
    "ECDH-SHA2-1.3.132.0.33 (nistp224, secp224r1)";
  reference
    "RFC 5656:"
```



```
        Elliptic Curve Algorithm Integration in the
        Secure Shell Transport Layer";
    }

    identity ecdh-sha2-1.3.132.0.26 {
        base key-exchange-alg-base;
        description
            "ECDH-SHA2-1.3.132.0.26 (nistk233, sect233k1)";
        reference
            "RFC 5656:
            Elliptic Curve Algorithm Integration in the
            Secure Shell Transport Layer";
    }

    identity ecdh-sha2-1.3.132.0.27 {
        base key-exchange-alg-base;
        description
            "ECDH-SHA2-1.3.132.0.27 (nistb233, sect233r1)";
        reference
            "RFC 5656:
            Elliptic Curve Algorithm Integration in the
            Secure Shell Transport Layer";
    }

    identity ecdh-sha2-1.3.132.0.16 {
        base key-exchange-alg-base;
        description
            "ECDH-SHA2-1.3.132.0.16 (nistk283, sect283k1)";
        reference
            "RFC 5656:
            Elliptic Curve Algorithm Integration in the
            Secure Shell Transport Layer";
    }

    identity ecdh-sha2-1.3.132.0.36 {
        base key-exchange-alg-base;
        description
            "ECDH-SHA2-1.3.132.0.36 (nistk409, sect409k1)";
        reference
            "RFC 5656:
            Elliptic Curve Algorithm Integration in the
            Secure Shell Transport Layer";
    }

    identity ecdh-sha2-1.3.132.0.37 {
        base key-exchange-alg-base;
        description
            "ECDH-SHA2-1.3.132.0.37 (nistb409, sect409r1)";
```

```
    reference
      "RFC 5656:
        Elliptic Curve Algorithm Integration in the
        Secure Shell Transport Layer";
  }

  identity ecdh-sha2-1.3.132.0.38 {
    base key-exchange-alg-base;
    description
      "ECDH-SHA2-1.3.132.0.38 (nistt571, sect571k1)";
    reference
      "RFC 5656:
        Elliptic Curve Algorithm Integration in the
        Secure Shell Transport Layer";
  }

  identity ecmqv-sha2 {
    base key-exchange-alg-base;
    description
      "ECMQV-SHA2";
    reference
      "RFC 5656:
        Elliptic Curve Algorithm Integration in the
        Secure Shell Transport Layer";
  }

  identity gss-group1-sha1-nistp256 {
    base key-exchange-alg-base;
    status deprecated;
    description
      "GSS-GROUP1-SHA1-NISTP256 (secp256r1)";
    reference
      "RFC 8732:
        Generic Security Service Application Program Interface
        (GSS-API) Key Exchange with SHA-2";
  }

  identity gss-group1-sha1-nistp384 {
    base key-exchange-alg-base;
    status deprecated;
    description
      "GSS-GROUP1-SHA1-NISTP384 (secp384r1)";
    reference
      "RFC 8732:
        Generic Security Service Application Program Interface
        (GSS-API) Key Exchange with SHA-2";
  }
```

```
identity gss-group1-sha1-nistp521 {
  base key-exchange-alg-base;
  status deprecated;
  description
    "GSS-GROUP1-SHA1-NISTP521 (secp521r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group1-sha1-1.3.132.0.1 {
  base key-exchange-alg-base;
  status deprecated;
  description
    "GSS-GROUP1-SHA1-1.3.132.0.1 (nistk163, sect163k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group1-sha1-1.2.840.10045.3.1.1 {
  base key-exchange-alg-base;
  status deprecated;
  description
    "GSS-GROUP1-SHA1-1.2.840.10045.3.1.1 (nistp192, secp192r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group1-sha1-1.3.132.0.33 {
  base key-exchange-alg-base;
  status deprecated;
  description
    "GSS-GROUP1-SHA1-1.3.132.0.33 (nistp224, secp224r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group1-sha1-1.3.132.0.26 {
  base key-exchange-alg-base;
  status deprecated;
  description
```

```
"GSS-GROUP1-SHA1-1.3.132.0.26 (nistk233, sect233k1)";
reference
  "RFC 8732:
    Generic Security Service Application Program Interface
    (GSS-API) Key Exchange with SHA-2";
}

identity gss-group1-sha1-1.3.132.0.27 {
  base key-exchange-alg-base;
  status deprecated;
  description
    "GSS-GROUP1-SHA1-1.3.132.0.27 (nistb233, sect233r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group1-sha1-1.3.132.0.16 {
  base key-exchange-alg-base;
  status deprecated;
  description
    "GSS-GROUP1-SHA1-1.3.132.0.16 (nistk283, sect283k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group1-sha1-1.3.132.0.36 {
  base key-exchange-alg-base;
  status deprecated;
  description
    "GSS-GROUP1-SHA1-1.3.132.0.36 (nistk409, sect409k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group1-sha1-1.3.132.0.37 {
  base key-exchange-alg-base;
  status deprecated;
  description
    "GSS-GROUP1-SHA1-1.3.132.0.37 (nistb409, sect409r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
```

```
        (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group1-sha1-1.3.132.0.38 {
        base key-exchange-alg-base;
        status deprecated;
        description
            "GSS-GROUP1-SHA1-1.3.132.0.38 (nistt571, sect571k1)";
        reference
            "RFC 8732:
             Generic Security Service Application Program Interface
             (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group1-sha1-curve25519-sha256 {
        base key-exchange-alg-base;
        status deprecated;
        description
            "GSS-GROUP1-SHA1-CURVE25519-SHA256";
        reference
            "RFC 8732:
             Generic Security Service Application Program Interface
             (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group1-sha1-curve448-sha512 {
        base key-exchange-alg-base;
        status deprecated;
        description
            "GSS-GROUP1-SHA1-CURVE448-SHA512";
        reference
            "RFC 8732:
             Generic Security Service Application Program Interface
             (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group14-sha1-nistp256 {
        base key-exchange-alg-base;
        status deprecated;
        description
            "GSS-GROUP14-SHA1-NISTP256 (secp256r1)";
        reference
            "RFC 8732:
             Generic Security Service Application Program Interface
             (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group14-sha1-nistp384 {
```

```
    base key-exchange-alg-base;
    status deprecated;
    description
        "GSS-GROUP14-SHA1-NISTP384 (secp384r1)";
    reference
        "RFC 8732:
        Generic Security Service Application Program Interface
        (GSS-API) Key Exchange with SHA-2";
}

identity gss-group14-sha1-nistp521 {
    base key-exchange-alg-base;
    status deprecated;
    description
        "GSS-GROUP14-SHA1-NISTP521 (secp521r1)";
    reference
        "RFC 8732:
        Generic Security Service Application Program Interface
        (GSS-API) Key Exchange with SHA-2";
}

identity gss-group14-sha1-1.3.132.0.1 {
    base key-exchange-alg-base;
    status deprecated;
    description
        "GSS-GROUP14-SHA1-1.3.132.0.1 (nistk163, sect163k1)";
    reference
        "RFC 8732:
        Generic Security Service Application Program Interface
        (GSS-API) Key Exchange with SHA-2";
}

identity gss-group14-sha1-1.2.840.10045.3.1.1 {
    base key-exchange-alg-base;
    status deprecated;
    description
        "GSS-GROUP14-SHA1-1.2.840.10045.3.1.1 (nistp192, secp192r1)";
    reference
        "RFC 8732:
        Generic Security Service Application Program Interface
        (GSS-API) Key Exchange with SHA-2";
}

identity gss-group14-sha1-1.3.132.0.33 {
    base key-exchange-alg-base;
    status deprecated;
    description
        "GSS-GROUP14-SHA1-1.3.132.0.33 (nistp224, secp224r1)";
```

```
reference
  "RFC 8732:
    Generic Security Service Application Program Interface
    (GSS-API) Key Exchange with SHA-2";
}

identity gss-group14-sha1-1.3.132.0.26 {
  base key-exchange-alg-base;
  status deprecated;
  description
    "GSS-GROUP14-SHA1-1.3.132.0.26 (nistk233, sect233k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group14-sha1-1.3.132.0.27 {
  base key-exchange-alg-base;
  status deprecated;
  description
    "GSS-GROUP14-SHA1-1.3.132.0.27 (nistb233, sect233r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group14-sha1-1.3.132.0.16 {
  base key-exchange-alg-base;
  status deprecated;
  description
    "GSS-GROUP14-SHA1-1.3.132.0.16 (nistk283, sect283k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group14-sha1-1.3.132.0.36 {
  base key-exchange-alg-base;
  status deprecated;
  description
    "GSS-GROUP14-SHA1-1.3.132.0.36 (nistk409, sect409k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
```

```
}

identity gss-group14-sha1-1.3.132.0.37 {
  base key-exchange-alg-base;
  status deprecated;
  description
    "GSS-GROUP14-SHA1-1.3.132.0.37 (nistb409, sect409r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group14-sha1-1.3.132.0.38 {
  base key-exchange-alg-base;
  status deprecated;
  description
    "GSS-GROUP14-SHA1-1.3.132.0.38 (nistt571, sect571k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group14-sha1-curve25519-sha256 {
  base key-exchange-alg-base;
  status deprecated;
  description
    "GSS-GROUP14-SHA1-CURVE25519-SHA256";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group14-sha1-curve448-sha512 {
  base key-exchange-alg-base;
  status deprecated;
  description
    "GSS-GROUP14-SHA1-CURVE448-SHA512";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-gex-sha1-nistp256 {
  base key-exchange-alg-base;
```



```
    status deprecated;
    description
        "GSS-GEX-SHA1-NISTP256 (secp256r1)";
    reference
        "RFC 8732:
        Generic Security Service Application Program Interface
        (GSS-API) Key Exchange with SHA-2";
}

identity gss-gex-sha1-nistp384 {
    base key-exchange-alg-base;
    status deprecated;
    description
        "GSS-GEX-SHA1-NISTP384 (secp384r1)";
    reference
        "RFC 8732:
        Generic Security Service Application Program Interface
        (GSS-API) Key Exchange with SHA-2";
}

identity gss-gex-sha1-nistp521 {
    base key-exchange-alg-base;
    status deprecated;
    description
        "GSS-GEX-SHA1-NISTP521 (secp521r1)";
    reference
        "RFC 8732:
        Generic Security Service Application Program Interface
        (GSS-API) Key Exchange with SHA-2";
}

identity gss-gex-sha1-1.3.132.0.1 {
    base key-exchange-alg-base;
    status deprecated;
    description
        "GSS-GEX-SHA1-1.3.132.0.1 (nistk163, sect163k1)";
    reference
        "RFC 8732:
        Generic Security Service Application Program Interface
        (GSS-API) Key Exchange with SHA-2";
}

identity gss-gex-sha1-1.2.840.10045.3.1.1 {
    base key-exchange-alg-base;
    status deprecated;
    description
        "GSS-GEX-SHA1-1.2.840.10045.3.1.1 (nistp192, secp192r1)";
    reference
```

```
"RFC 8732:
    Generic Security Service Application Program Interface
    (GSS-API) Key Exchange with SHA-2";
}

identity gss-gex-sha1-1.3.132.0.33 {
    base key-exchange-alg-base;
    status deprecated;
    description
        "GSS-GEX-SHA1-1.3.132.0.33 (nistp224, secp224r1)";
    reference
        "RFC 8732:
            Generic Security Service Application Program Interface
            (GSS-API) Key Exchange with SHA-2";
}

identity gss-gex-sha1-1.3.132.0.26 {
    base key-exchange-alg-base;
    status deprecated;
    description
        "GSS-GEX-SHA1-1.3.132.0.26 (nistk233, sect233k1)";
    reference
        "RFC 8732:
            Generic Security Service Application Program Interface
            (GSS-API) Key Exchange with SHA-2";
}

identity gss-gex-sha1-1.3.132.0.27 {
    base key-exchange-alg-base;
    status deprecated;
    description
        "GSS-GEX-SHA1-1.3.132.0.27 (nistb233, sect233r1)";
    reference
        "RFC 8732:
            Generic Security Service Application Program Interface
            (GSS-API) Key Exchange with SHA-2";
}

identity gss-gex-sha1-1.3.132.0.16 {
    base key-exchange-alg-base;
    status deprecated;
    description
        "GSS-GEX-SHA1-1.3.132.0.16 (nistk283, sect283k1)";
    reference
        "RFC 8732:
            Generic Security Service Application Program Interface
            (GSS-API) Key Exchange with SHA-2";
}
```

```
identity gss-gex-sha1-1.3.132.0.36 {
  base key-exchange-alg-base;
  status deprecated;
  description
    "GSS-GEX-SHA1-1.3.132.0.36 (nistk409, sect409k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-gex-sha1-1.3.132.0.37 {
  base key-exchange-alg-base;
  status deprecated;
  description
    "GSS-GEX-SHA1-1.3.132.0.37 (nistb409, sect409r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-gex-sha1-1.3.132.0.38 {
  base key-exchange-alg-base;
  status deprecated;
  description
    "GSS-GEX-SHA1-1.3.132.0.38 (nistt571, sect571k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-gex-sha1-curve25519-sha256 {
  base key-exchange-alg-base;
  status deprecated;
  description
    "GSS-GEX-SHA1-CURVE25519-SHA256";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-gex-sha1-curve448-sha512 {
  base key-exchange-alg-base;
  status deprecated;
  description
```

```
    "GSS-GEX-SHA1-CURVE448-SHA512";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity rsa1024-sha1 {
  base key-exchange-alg-base;
  description
    "RSA1024-SHA1";
  reference
    "RFC 4432:
      RSA Key Exchange for the Secure Shell (SSH)
      Transport Layer Protocol";
}

identity rsa2048-sha256 {
  base key-exchange-alg-base;
  description
    "RSA2048-SHA256";
  reference
    "RFC 4432:
      RSA Key Exchange for the Secure Shell (SSH)
      Transport Layer Protocol";
}

identity ext-info-s {
  base key-exchange-alg-base;
  description
    "EXT-INFO-S";
  reference
    "RFC 8308:
      Extension Negotiation in the Secure Shell (SSH) Protocol";
}

identity ext-info-c {
  base key-exchange-alg-base;
  description
    "EXT-INFO-C";
  reference
    "RFC 8308:
      Extension Negotiation in the Secure Shell (SSH) Protocol";
}

identity gss-group14-sha256-nistp256 {
  base key-exchange-alg-base;
  description
```

```
    "GSS-GROUP14-SHA256-NISTP256 (secp256r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group14-sha256-nistp384 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP14-SHA256-NISTP384 (secp384r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group14-sha256-nistp521 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP14-SHA256-NISTP521 (secp521r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group14-sha256-1.3.132.0.1 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP14-SHA256-1.3.132.0.1 (nistk163, sect163k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group14-sha256-1.2.840.10045.3.1.1 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP14-SHA256-1.2.840.10045.3.1.1 (nistp192, secp192r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group14-sha256-1.3.132.0.33 {
```

```
    base key-exchange-alg-base;
    description
        "GSS-GROUP14-SHA256-1.3.132.0.33 (nistp224, secp224r1)";
    reference
        "RFC 8732:
        Generic Security Service Application Program Interface
        (GSS-API) Key Exchange with SHA-2";
}

identity gss-group14-sha256-1.3.132.0.26 {
    base key-exchange-alg-base;
    description
        "GSS-GROUP14-SHA256-1.3.132.0.26 (nistk233, sect233k1)";
    reference
        "RFC 8732:
        Generic Security Service Application Program Interface
        (GSS-API) Key Exchange with SHA-2";
}

identity gss-group14-sha256-1.3.132.0.27 {
    base key-exchange-alg-base;
    description
        "GSS-GROUP14-SHA256-1.3.132.0.27 (nistb233, sect233r1)";
    reference
        "RFC 8732:
        Generic Security Service Application Program Interface
        (GSS-API) Key Exchange with SHA-2";
}

identity gss-group14-sha256-1.3.132.0.16 {
    base key-exchange-alg-base;
    description
        "GSS-GROUP14-SHA256-1.3.132.0.16 (nistk283, sect283k1)";
    reference
        "RFC 8732:
        Generic Security Service Application Program Interface
        (GSS-API) Key Exchange with SHA-2";
}

identity gss-group14-sha256-1.3.132.0.36 {
    base key-exchange-alg-base;
    description
        "GSS-GROUP14-SHA256-1.3.132.0.36 (nistk409, sect409k1)";
    reference
        "RFC 8732:
        Generic Security Service Application Program Interface
        (GSS-API) Key Exchange with SHA-2";
}
```

```
identity gss-group14-sha256-1.3.132.0.37 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP14-SHA256-1.3.132.0.37 (nistb409, sect409r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group14-sha256-1.3.132.0.38 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP14-SHA256-1.3.132.0.38 (nistt571, sect571k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group14-sha256-curve25519-sha256 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP14-SHA256-CURVE25519-SHA256";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group14-sha256-curve448-sha512 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP14-SHA256-CURVE448-SHA512";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group15-sha512-nistp256 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP15-SHA512-NISTP256 (secp256r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}
```

```
}

identity gss-group15-sha512-nistp384 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP15-SHA512-NISTP384 (secp384r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group15-sha512-nistp521 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP15-SHA512-NISTP521 (secp521r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group15-sha512-1.3.132.0.1 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP15-SHA512-1.3.132.0.1 (nistk163, sect163k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group15-sha512-1.2.840.10045.3.1.1 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP15-SHA512-1.2.840.10045.3.1.1 (nistp192, secp192r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group15-sha512-1.3.132.0.33 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP15-SHA512-1.3.132.0.33 (nistp224, secp224r1)";
  reference
    "RFC 8732:"
```



```
        Generic Security Service Application Program Interface
        (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group15-sha512-1.3.132.0.26 {
        base key-exchange-alg-base;
        description
            "GSS-GROUP15-SHA512-1.3.132.0.26 (nistk233, sect233k1)";
        reference
            "RFC 8732:
             Generic Security Service Application Program Interface
             (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group15-sha512-1.3.132.0.27 {
        base key-exchange-alg-base;
        description
            "GSS-GROUP15-SHA512-1.3.132.0.27 (nistb233, sect233r1)";
        reference
            "RFC 8732:
             Generic Security Service Application Program Interface
             (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group15-sha512-1.3.132.0.16 {
        base key-exchange-alg-base;
        description
            "GSS-GROUP15-SHA512-1.3.132.0.16 (nistk283, sect283k1)";
        reference
            "RFC 8732:
             Generic Security Service Application Program Interface
             (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group15-sha512-1.3.132.0.36 {
        base key-exchange-alg-base;
        description
            "GSS-GROUP15-SHA512-1.3.132.0.36 (nistk409, sect409k1)";
        reference
            "RFC 8732:
             Generic Security Service Application Program Interface
             (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group15-sha512-1.3.132.0.37 {
        base key-exchange-alg-base;
        description
            "GSS-GROUP15-SHA512-1.3.132.0.37 (nistb409, sect409r1)";
```

```
reference
  "RFC 8732:
    Generic Security Service Application Program Interface
    (GSS-API) Key Exchange with SHA-2";
}

identity gss-group15-sha512-1.3.132.0.38 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP15-SHA512-1.3.132.0.38 (nistt571, sect571k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group15-sha512-curve25519-sha256 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP15-SHA512-CURVE25519-SHA256";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group15-sha512-curve448-sha512 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP15-SHA512-CURVE448-SHA512";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group16-sha512-nistp256 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP16-SHA512-NISTP256 (secp256r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group16-sha512-nistp384 {
  base key-exchange-alg-base;
```

```
description
  "GSS-GROUP16-SHA512-NISTP384 (secp384r1)";
reference
  "RFC 8732:
    Generic Security Service Application Program Interface
    (GSS-API) Key Exchange with SHA-2";
}

identity gss-group16-sha512-nistp521 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP16-SHA512-NISTP521 (secp521r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group16-sha512-1.3.132.0.1 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP16-SHA512-1.3.132.0.1 (nistk163, sect163k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group16-sha512-1.2.840.10045.3.1.1 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP16-SHA512-1.2.840.10045.3.1.1 (nistp192, secp192r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group16-sha512-1.3.132.0.33 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP16-SHA512-1.3.132.0.33 (nistp224, secp224r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}
```

```
identity gss-group16-sha512-1.3.132.0.26 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP16-SHA512-1.3.132.0.26 (nistk233, sect233k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group16-sha512-1.3.132.0.27 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP16-SHA512-1.3.132.0.27 (nistb233, sect233r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group16-sha512-1.3.132.0.16 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP16-SHA512-1.3.132.0.16 (nistk283, sect283k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group16-sha512-1.3.132.0.36 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP16-SHA512-1.3.132.0.36 (nistk409, sect409k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group16-sha512-1.3.132.0.37 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP16-SHA512-1.3.132.0.37 (nistb409, sect409r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}
```

```
}

identity gss-group16-sha512-1.3.132.0.38 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP16-SHA512-1.3.132.0.38 (nistt571, sect571k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group16-sha512-curve25519-sha256 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP16-SHA512-CURVE25519-SHA256";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group16-sha512-curve448-sha512 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP16-SHA512-CURVE448-SHA512";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group17-sha512-nistp256 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP17-SHA512-NISTP256 (secp256r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group17-sha512-nistp384 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP17-SHA512-NISTP384 (secp384r1)";
  reference
    "RFC 8732:"
```

```
        Generic Security Service Application Program Interface
        (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group17-sha512-nistp521 {
        base key-exchange-alg-base;
        description
            "GSS-GROUP17-SHA512-NISTP521 (secp521r1)";
        reference
            "RFC 8732:
            Generic Security Service Application Program Interface
            (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group17-sha512-1.3.132.0.1 {
        base key-exchange-alg-base;
        description
            "GSS-GROUP17-SHA512-1.3.132.0.1 (nistk163, sect163k1)";
        reference
            "RFC 8732:
            Generic Security Service Application Program Interface
            (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group17-sha512-1.2.840.10045.3.1.1 {
        base key-exchange-alg-base;
        description
            "GSS-GROUP17-SHA512-1.2.840.10045.3.1.1 (nistp192, secp192r1)";
        reference
            "RFC 8732:
            Generic Security Service Application Program Interface
            (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group17-sha512-1.3.132.0.33 {
        base key-exchange-alg-base;
        description
            "GSS-GROUP17-SHA512-1.3.132.0.33 (nistp224, secp224r1)";
        reference
            "RFC 8732:
            Generic Security Service Application Program Interface
            (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group17-sha512-1.3.132.0.26 {
        base key-exchange-alg-base;
        description
            "GSS-GROUP17-SHA512-1.3.132.0.26 (nistk233, sect233k1)";
```

```
reference
  "RFC 8732:
    Generic Security Service Application Program Interface
    (GSS-API) Key Exchange with SHA-2";
}

identity gss-group17-sha512-1.3.132.0.27 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP17-SHA512-1.3.132.0.27 (nistb233, sect233r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group17-sha512-1.3.132.0.16 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP17-SHA512-1.3.132.0.16 (nistk283, sect283k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group17-sha512-1.3.132.0.36 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP17-SHA512-1.3.132.0.36 (nistk409, sect409k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group17-sha512-1.3.132.0.37 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP17-SHA512-1.3.132.0.37 (nistb409, sect409r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group17-sha512-1.3.132.0.38 {
  base key-exchange-alg-base;
```

```
description
  "GSS-GROUP17-SHA512-1.3.132.0.38 (nistt571, sect571k1)";
reference
  "RFC 8732:
    Generic Security Service Application Program Interface
    (GSS-API) Key Exchange with SHA-2";
}

identity gss-group17-sha512-curve25519-sha256 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP17-SHA512-CURVE25519-SHA256";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group17-sha512-curve448-sha512 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP17-SHA512-CURVE448-SHA512";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group18-sha512-nistp256 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP18-SHA512-NISTP256 (secp256r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group18-sha512-nistp384 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP18-SHA512-NISTP384 (secp384r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}
```



```
identity gss-group18-sha512-nistp521 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP18-SHA512-NISTP521 (secp521r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group18-sha512-1.3.132.0.1 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP18-SHA512-1.3.132.0.1 (nistk163, sect163k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group18-sha512-1.2.840.10045.3.1.1 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP18-SHA512-1.2.840.10045.3.1.1 (nistp192, secp192r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group18-sha512-1.3.132.0.33 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP18-SHA512-1.3.132.0.33 (nistp224, secp224r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group18-sha512-1.3.132.0.26 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP18-SHA512-1.3.132.0.26 (nistk233, sect233k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}
```

```
}

identity gss-group18-sha512-1.3.132.0.27 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP18-SHA512-1.3.132.0.27 (nistb233, sect233r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group18-sha512-1.3.132.0.16 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP18-SHA512-1.3.132.0.16 (nistk283, sect283k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group18-sha512-1.3.132.0.36 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP18-SHA512-1.3.132.0.36 (nistk409, sect409k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group18-sha512-1.3.132.0.37 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP18-SHA512-1.3.132.0.37 (nistb409, sect409r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-group18-sha512-1.3.132.0.38 {
  base key-exchange-alg-base;
  description
    "GSS-GROUP18-SHA512-1.3.132.0.38 (nistt571, sect571k1)";
  reference
    "RFC 8732:
```

```
        Generic Security Service Application Program Interface
        (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group18-sha512-curve25519-sha256 {
        base key-exchange-alg-base;
        description
            "GSS-GROUP18-SHA512-CURVE25519-SHA256";
        reference
            "RFC 8732:
            Generic Security Service Application Program Interface
            (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-group18-sha512-curve448-sha512 {
        base key-exchange-alg-base;
        description
            "GSS-GROUP18-SHA512-CURVE448-SHA512";
        reference
            "RFC 8732:
            Generic Security Service Application Program Interface
            (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-nistp256-sha256-nistp256 {
        base key-exchange-alg-base;
        description
            "GSS-NISTP256-SHA256-NISTP256 (secp256r1)";
        reference
            "RFC 8732:
            Generic Security Service Application Program Interface
            (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-nistp256-sha256-nistp384 {
        base key-exchange-alg-base;
        description
            "GSS-NISTP256-SHA256-NISTP384 (secp384r1)";
        reference
            "RFC 8732:
            Generic Security Service Application Program Interface
            (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-nistp256-sha256-nistp521 {
        base key-exchange-alg-base;
        description
            "GSS-NISTP256-SHA256-NISTP521 (secp521r1)";
```

```
reference
  "RFC 8732:
    Generic Security Service Application Program Interface
    (GSS-API) Key Exchange with SHA-2";
}

identity gss-nistp256-sha256-1.3.132.0.1 {
  base key-exchange-alg-base;
  description
    "GSS-NISTP256-SHA256-1.3.132.0.1 (nistk163, sect163k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-nistp256-sha256-1.2.840.10045.3.1.1 {
  base key-exchange-alg-base;
  description
    "GSS-NISTP256-SHA256-1.2.840.10045.3.1.1 (nistp192, secp192r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-nistp256-sha256-1.3.132.0.33 {
  base key-exchange-alg-base;
  description
    "GSS-NISTP256-SHA256-1.3.132.0.33 (nistp224, secp224r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-nistp256-sha256-1.3.132.0.26 {
  base key-exchange-alg-base;
  description
    "GSS-NISTP256-SHA256-1.3.132.0.26 (nistk233, sect233k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-nistp256-sha256-1.3.132.0.27 {
  base key-exchange-alg-base;
```

```
description
  "GSS-NISTP256-SHA256-1.3.132.0.27 (nistb233, sect233r1)";
reference
  "RFC 8732:
    Generic Security Service Application Program Interface
    (GSS-API) Key Exchange with SHA-2";
}

identity gss-nistp256-sha256-1.3.132.0.16 {
  base key-exchange-alg-base;
  description
    "GSS-NISTP256-SHA256-1.3.132.0.16 (nistk283, sect283k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-nistp256-sha256-1.3.132.0.36 {
  base key-exchange-alg-base;
  description
    "GSS-NISTP256-SHA256-1.3.132.0.36 (nistk409, sect409k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-nistp256-sha256-1.3.132.0.37 {
  base key-exchange-alg-base;
  description
    "GSS-NISTP256-SHA256-1.3.132.0.37 (nistb409, sect409r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-nistp256-sha256-1.3.132.0.38 {
  base key-exchange-alg-base;
  description
    "GSS-NISTP256-SHA256-1.3.132.0.38 (nistt571, sect571k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}
```

```
identity gss-nistp256-sha256-curve25519-sha256 {
  base key-exchange-alg-base;
  description
    "GSS-NISTP256-SHA256-CURVE25519-SHA256";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-nistp256-sha256-curve448-sha512 {
  base key-exchange-alg-base;
  description
    "GSS-NISTP256-SHA256-CURVE448-SHA512";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-nistp384-sha384-nistp256 {
  base key-exchange-alg-base;
  description
    "GSS-NISTP384-SHA384-NISTP256 (secp256r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-nistp384-sha384-nistp384 {
  base key-exchange-alg-base;
  description
    "GSS-NISTP384-SHA384-NISTP384 (secp384r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-nistp384-sha384-nistp521 {
  base key-exchange-alg-base;
  description
    "GSS-NISTP384-SHA384-NISTP521 (secp521r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}
```

```
}

identity gss-nistp384-sha384-1.3.132.0.1 {
  base key-exchange-alg-base;
  description
    "GSS-NISTP384-SHA384-1.3.132.0.1 (nistk163, sect163k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-nistp384-sha384-1.2.840.10045.3.1.1 {
  base key-exchange-alg-base;
  description
    "GSS-NISTP384-SHA384-1.2.840.10045.3.1.1 (nistp192, secp192r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-nistp384-sha384-1.3.132.0.33 {
  base key-exchange-alg-base;
  description
    "GSS-NISTP384-SHA384-1.3.132.0.33 (nistp224, secp224r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-nistp384-sha384-1.3.132.0.26 {
  base key-exchange-alg-base;
  description
    "GSS-NISTP384-SHA384-1.3.132.0.26 (nistk233, sect233k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-nistp384-sha384-1.3.132.0.27 {
  base key-exchange-alg-base;
  description
    "GSS-NISTP384-SHA384-1.3.132.0.27 (nistb233, sect233r1)";
  reference
    "RFC 8732:
```

```
        Generic Security Service Application Program Interface
        (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-nistp384-sha384-1.3.132.0.16 {
        base key-exchange-alg-base;
        description
            "GSS-NISTP384-SHA384-1.3.132.0.16 (nistk283, sect283k1)";
        reference
            "RFC 8732:
            Generic Security Service Application Program Interface
            (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-nistp384-sha384-1.3.132.0.36 {
        base key-exchange-alg-base;
        description
            "GSS-NISTP384-SHA384-1.3.132.0.36 (nistk409, sect409k1)";
        reference
            "RFC 8732:
            Generic Security Service Application Program Interface
            (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-nistp384-sha384-1.3.132.0.37 {
        base key-exchange-alg-base;
        description
            "GSS-NISTP384-SHA384-1.3.132.0.37 (nistb409, sect409r1)";
        reference
            "RFC 8732:
            Generic Security Service Application Program Interface
            (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-nistp384-sha384-1.3.132.0.38 {
        base key-exchange-alg-base;
        description
            "GSS-NISTP384-SHA384-1.3.132.0.38 (nistt571, sect571k1)";
        reference
            "RFC 8732:
            Generic Security Service Application Program Interface
            (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-nistp384-sha384-curve25519-sha256 {
        base key-exchange-alg-base;
        description
            "GSS-NISTP384-SHA384-CURVE25519-SHA256";
    }
```



```
reference
  "RFC 8732:
    Generic Security Service Application Program Interface
    (GSS-API) Key Exchange with SHA-2";
}

identity gss-nistp384-sha384-curve448-sha512 {
  base key-exchange-alg-base;
  description
    "GSS-NISTP384-SHA384-CURVE448-SHA512";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-nistp521-sha512-nistp256 {
  base key-exchange-alg-base;
  description
    "GSS-NISTP521-SHA512-NISTP256 (secp256r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-nistp521-sha512-nistp384 {
  base key-exchange-alg-base;
  description
    "GSS-NISTP521-SHA512-NISTP384 (secp384r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-nistp521-sha512-nistp521 {
  base key-exchange-alg-base;
  description
    "GSS-NISTP521-SHA512-NISTP521 (secp521r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-nistp521-sha512-1.3.132.0.1 {
  base key-exchange-alg-base;
```

```
description
  "GSS-NISTP521-SHA512-1.3.132.0.1 (nistk163, sect163k1)";
reference
  "RFC 8732:
    Generic Security Service Application Program Interface
    (GSS-API) Key Exchange with SHA-2";
}

identity gss-nistp521-sha512-1.2.840.10045.3.1.1 {
  base key-exchange-alg-base;
  description
    "GSS-NISTP521-SHA512-1.2.840.10045.3.1.1 (nistp192, secp192r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-nistp521-sha512-1.3.132.0.33 {
  base key-exchange-alg-base;
  description
    "GSS-NISTP521-SHA512-1.3.132.0.33 (nistp224, secp224r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-nistp521-sha512-1.3.132.0.26 {
  base key-exchange-alg-base;
  description
    "GSS-NISTP521-SHA512-1.3.132.0.26 (nistk233, sect233k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-nistp521-sha512-1.3.132.0.27 {
  base key-exchange-alg-base;
  description
    "GSS-NISTP521-SHA512-1.3.132.0.27 (nistb233, sect233r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}
```

```
identity gss-nistp521-sha512-1.3.132.0.16 {
  base key-exchange-alg-base;
  description
    "GSS-NISTP521-SHA512-1.3.132.0.16 (nistk283, sect283k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-nistp521-sha512-1.3.132.0.36 {
  base key-exchange-alg-base;
  description
    "GSS-NISTP521-SHA512-1.3.132.0.36 (nistk409, sect409k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-nistp521-sha512-1.3.132.0.37 {
  base key-exchange-alg-base;
  description
    "GSS-NISTP521-SHA512-1.3.132.0.37 (nistb409, sect409r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-nistp521-sha512-1.3.132.0.38 {
  base key-exchange-alg-base;
  description
    "GSS-NISTP521-SHA512-1.3.132.0.38 (nistt571, sect571k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-nistp521-sha512-curve25519-sha256 {
  base key-exchange-alg-base;
  description
    "GSS-NISTP521-SHA512-CURVE25519-SHA256";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}
```

```
}

identity gss-nistp521-sha512-curve448-sha512 {
  base key-exchange-alg-base;
  description
    "GSS-NISTP521-SHA512-CURVE448-SHA512";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-curve25519-sha256-nistp256 {
  base key-exchange-alg-base;
  description
    "GSS-CURVE25519-SHA256-NISTP256 (secp256r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-curve25519-sha256-nistp384 {
  base key-exchange-alg-base;
  description
    "GSS-CURVE25519-SHA256-NISTP384 (secp384r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-curve25519-sha256-nistp521 {
  base key-exchange-alg-base;
  description
    "GSS-CURVE25519-SHA256-NISTP521 (secp521r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-curve25519-sha256-1.3.132.0.1 {
  base key-exchange-alg-base;
  description
    "GSS-CURVE25519-SHA256-1.3.132.0.1 (nistk163, sect163k1)";
  reference
    "RFC 8732:
```

```
        Generic Security Service Application Program Interface
        (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-curve25519-sha256-1.2.840.10045.3.1.1 {
        base key-exchange-alg-base;
        description
            "GSS-CURVE25519-SHA256-1.2.840.10045.3.1.1 (nistp192,
            secp192r1)";
        reference
            "RFC 8732:
            Generic Security Service Application Program Interface
            (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-curve25519-sha256-1.3.132.0.33 {
        base key-exchange-alg-base;
        description
            "GSS-CURVE25519-SHA256-1.3.132.0.33 (nistp224, secp224r1)";
        reference
            "RFC 8732:
            Generic Security Service Application Program Interface
            (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-curve25519-sha256-1.3.132.0.26 {
        base key-exchange-alg-base;
        description
            "GSS-CURVE25519-SHA256-1.3.132.0.26 (nistk233, sect233k1)";
        reference
            "RFC 8732:
            Generic Security Service Application Program Interface
            (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-curve25519-sha256-1.3.132.0.27 {
        base key-exchange-alg-base;
        description
            "GSS-CURVE25519-SHA256-1.3.132.0.27 (nistb233, sect233r1)";
        reference
            "RFC 8732:
            Generic Security Service Application Program Interface
            (GSS-API) Key Exchange with SHA-2";
    }

    identity gss-curve25519-sha256-1.3.132.0.16 {
        base key-exchange-alg-base;
        description
```

```
    "GSS-CURVE25519-SHA256-1.3.132.0.16 (nistk283, sect283k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-curve25519-sha256-1.3.132.0.36 {
  base key-exchange-alg-base;
  description
    "GSS-CURVE25519-SHA256-1.3.132.0.36 (nistk409, sect409k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-curve25519-sha256-1.3.132.0.37 {
  base key-exchange-alg-base;
  description
    "GSS-CURVE25519-SHA256-1.3.132.0.37 (nistb409, sect409r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-curve25519-sha256-1.3.132.0.38 {
  base key-exchange-alg-base;
  description
    "GSS-CURVE25519-SHA256-1.3.132.0.38 (nistt571, sect571k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-curve25519-sha256-curve25519-sha256 {
  base key-exchange-alg-base;
  description
    "GSS-CURVE25519-SHA256-CURVE25519-SHA256";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-curve25519-sha256-curve448-sha512 {
```

```
    base key-exchange-alg-base;
    description
        "GSS-CURVE25519-SHA256-CURVE448-SHA512";
    reference
        "RFC 8732:
        Generic Security Service Application Program Interface
        (GSS-API) Key Exchange with SHA-2";
}

identity gss-curve448-sha512-nistp256 {
    base key-exchange-alg-base;
    description
        "GSS-CURVE448-SHA512-NISTP256 (secp256r1)";
    reference
        "RFC 8732:
        Generic Security Service Application Program Interface
        (GSS-API) Key Exchange with SHA-2";
}

identity gss-curve448-sha512-nistp384 {
    base key-exchange-alg-base;
    description
        "GSS-CURVE448-SHA512-NISTP384 (secp384r1)";
    reference
        "RFC 8732:
        Generic Security Service Application Program Interface
        (GSS-API) Key Exchange with SHA-2";
}

identity gss-curve448-sha512-nistp521 {
    base key-exchange-alg-base;
    description
        "GSS-CURVE448-SHA512-NISTP521 (secp521r1)";
    reference
        "RFC 8732:
        Generic Security Service Application Program Interface
        (GSS-API) Key Exchange with SHA-2";
}

identity gss-curve448-sha512-1.3.132.0.1 {
    base key-exchange-alg-base;
    description
        "GSS-CURVE448-SHA512-1.3.132.0.1 (nistk163, sect163k1)";
    reference
        "RFC 8732:
        Generic Security Service Application Program Interface
        (GSS-API) Key Exchange with SHA-2";
}
```

```
identity gss-curve448-sha512-1.2.840.10045.3.1.1 {
  base key-exchange-alg-base;
  description
    "GSS-CURVE448-SHA512-1.2.840.10045.3.1.1 (nistp192, secp192r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-curve448-sha512-1.3.132.0.33 {
  base key-exchange-alg-base;
  description
    "GSS-CURVE448-SHA512-1.3.132.0.33 (nistp224, secp224r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-curve448-sha512-1.3.132.0.26 {
  base key-exchange-alg-base;
  description
    "GSS-CURVE448-SHA512-1.3.132.0.26 (nistk233, sect233k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-curve448-sha512-1.3.132.0.27 {
  base key-exchange-alg-base;
  description
    "GSS-CURVE448-SHA512-1.3.132.0.27 (nistb233, sect233r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-curve448-sha512-1.3.132.0.16 {
  base key-exchange-alg-base;
  description
    "GSS-CURVE448-SHA512-1.3.132.0.16 (nistk283, sect283k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}
```



```
}

identity gss-curve448-sha512-1.3.132.0.36 {
  base key-exchange-alg-base;
  description
    "GSS-CURVE448-SHA512-1.3.132.0.36 (nistk409, sect409k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-curve448-sha512-1.3.132.0.37 {
  base key-exchange-alg-base;
  description
    "GSS-CURVE448-SHA512-1.3.132.0.37 (nistb409, sect409r1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-curve448-sha512-1.3.132.0.38 {
  base key-exchange-alg-base;
  description
    "GSS-CURVE448-SHA512-1.3.132.0.38 (nistt571, sect571k1)";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-curve448-sha512-curve25519-sha256 {
  base key-exchange-alg-base;
  description
    "GSS-CURVE448-SHA512-CURVE25519-SHA256";
  reference
    "RFC 8732:
      Generic Security Service Application Program Interface
      (GSS-API) Key Exchange with SHA-2";
}

identity gss-curve448-sha512-curve448-sha512 {
  base key-exchange-alg-base;
  description
    "GSS-CURVE448-SHA512-CURVE448-SHA512";
  reference
    "RFC 8732:
```

```
        Generic Security Service Application Program Interface
        (GSS-API) Key Exchange with SHA-2";
    }

    identity curve25519-sha256 {
        base key-exchange-alg-base;
        description
            "CURVE25519-SHA256";
        reference
            "RFC 8731:
            Secure Shell (SSH) Key Exchange Method
            Using Curve25519 and Curve448";
    }

    identity curve448-sha512 {
        base key-exchange-alg-base;
        description
            "CURVE448-SHA512";
        reference
            "RFC 8731:
            Secure Shell (SSH) Key Exchange Method
            Using Curve25519 and Curve448";
    }

    // Protocol-accessible Nodes

    container supported-algorithms {
        config false;
        description
            "A container for a list of key exchange algorithms
            supported by the server.";
        leaf-list supported-algorithm {
            type key-exchange-algorithm-ref;
            description
                "A key exchange algorithm supported by the server.";
        }
    }
}

<CODE ENDS>
```

## Appendix B. Change Log

This section is to be removed before publishing as an RFC.

### B.1. 00 to 01

- \* Noted that '0.0.0.0' and ':::' might have special meanings.

- \* Renamed "keychain" to "keystore".

#### B.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.

#### B.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'.

#### B.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)

#### B.5. 04 to 05

- \* Merged changes from co-author.

#### B.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.

#### B.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.

#### B.8. 07 to 08

- \* Added a number of compatibility matrices to Section 5 (thanks Frank!)
- \* Clarified that any configured "host-key-alg" values need to be compatible with the configured private key.

#### B.9. 08 to 09

- \* Updated examples to reflect update to groupings defined in the keystore -09 draft.
- \* Add SSH keepalives features and groupings.
- \* Prefixed top-level SSH grouping nodes with 'ssh-' and support mashups.
- \* Updated copyright date, boilerplate template, affiliation, and folding algorithm.

## B.10. 09 to 10

- \* Reformatted the YANG modules.

## B.11. 10 to 11

- \* Reformatted lines causing folding to occur.

## B.12. 11 to 12

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

## B.13. 12 to 13

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

## B.14. 13 to 14

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

## B.15. 14 to 15

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

- \* Updated "server-authentication" and "client-authentication" nodes from being a leaf of type "ts:host-keys-ref" or "ts:certificates-ref" to a container that uses "ts:local-or-truststore-host-keys-grouping" or "ts:local-or-truststore-certs-grouping".

## B.16. 15 to 16

- \* Removed unnecessary if-feature statements in the -client and -server modules.
- \* Cleaned up some description statements in the -client and -server modules.
- \* Fixed a canonical ordering issue in ietf-ssh-common detected by new pyang.

## B.17. 16 to 17

- \* Removed choice local-or-external by removing the 'external' case and flattening the 'local' case and adding a "local-users-supported" feature.
- \* Updated examples to include the "\*-key-format" nodes.
- \* Augmented-in "must" expressions ensuring that locally-defined public-key-format are "ct:ssh-public-key-format" (must expr for ref'ed keys are TBD).

## B.18. 17 to 18

- \* Removed leaf-list 'other' from ietf-ssh-server.
- \* Removed unused 'external-client-auth-supported' feature.
- \* Added features client-auth-password, client-auth-hostbased, and client-auth-none.
- \* Renamed 'host-key' to 'public-key' for when refering to 'publickey' based auth.
- \* Added new feature-protected 'hostbased' and 'none' to the 'user' node's config.
- \* Added new feature-protected 'hostbased' and 'none' to the 'client-identity' node's config.
- \* Updated examples to reflect new "bag" addition to truststore.

- \* Refined truststore/keystore groupings to ensure the key formats "must" be particular values.
- \* Switched to using truststore's new "public-key" bag (instead of separate "ssh-public-key" and "raw-public-key" bags).
- \* Updated client/server examples to cover ALL cases (local/ref x cert/raw-key/psk).

#### B.19. 18 to 19

- \* Updated the "keepalives" containers to address Michal Vasko's request to align with RFC 8071.
- \* Removed algorithm-mapping tables from the "SSH Common Model" section
- \* Removed 'algorithm' node from examples.
- \* Added feature "userauth-publickey"
- \* Removed "choice auth-type", as auth-types are not exclusive.
- \* Renamed both "client-certs" and "server-certs" to "ee-certs"
- \* Switch "must" to assert the public-key-format is "subject-public-key-info-format" when certificates are used.
- \* Added a "Note to Reviewers" note to first page.

#### B.20. 19 to 20

- \* Added a "must 'public-key or password or hostbased or none or certificate'" statement to the "user" node in ietf-ssh-client
- \* Expanded "Data Model Overview section(s) [remove "wall" of tree diagrams].
- \* Moved the "ietf-ssh-common" module section to proceed the other two module sections.
- \* Updated the Security Considerations section.

#### B.21. 20 to 21

- \* Updated examples to reflect new "cleartext-" prefix in the crypto-types draft.

## B.22. 21 to 22

- \* Cleaned up the SSH-client examples (i.e., removing FIXMEs)
- \* Fixed issues found by the SecDir review of the "keystore" draft.
- \* Updated the "ietf-ssh-client" module to use the new "password-grouping" grouping from the "crypto-types" module.

## B.23. 22 to 23

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

## B.24. 23 to 24

- \* Removed the 'supported-authentication-methods' from {grouping ssh-server-grouping}/client-authentication.
- \* Added XML-comment above examples explaining the reason for the unexpected top-most element's presence.
- \* Added RFC-references to various 'feature' statements.
- \* Renamed "credentials" to "authentication methods"
- \* Renamed "client-auth-\*" to "userauth-\*
- \* Renamed "client-identity-\*" to "userauth-\*
- \* Fixed nits found by YANG Doctor reviews.
- \* Aligned modules with 'pyang -f' formatting.
- \* Added a 'Contributors' section.

## B.25. 24 to 25

- \* Moved algorithms in ietf-ssh-common (plus more) to IANA-maintained modules
- \* Added "config false" lists for algorithms supported by the server.
- \* Renamed "{ietf-ssh-client}userauth-\*" to "client-ident-\*
- \* Renamed "{ietf-ssh-server}userauth-\*" to "local-user-auth-\*
- \* Fixed issues found during YANG Doctor review.



- \* Fixed issues found during Secdir review.

## B.26. 25 to 26

- \* Replaced "base64encodedvalue==" with "BASE64VALUE=" in examples.
- \* Minor editorial nits

## B.27. 26 to 27

- \* Fixed up the 'WG Web' and 'WG List' lines in YANG module(s)
- \* Fixed up copyright (i.e., s/Simplified/Revised/) in YANG module(s)
- \* Created identityref-based typedefs for each of the four IANA alg identity bases.
- \* Added ietf-ssh-common:generate-public-key() RPC for discussion.

## Acknowledgements

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

## Contributors

Special acknowledgement goes to Gary Wu for his work on the "ietf-ssh-common" module.

## Author's Address

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

NETCONF  
Internet-Draft  
Intended status: Standards Track  
Expires: August 27, 2017

E. Voit  
Cisco Systems  
A. Clemm  
Huawei  
A. Gonzalez Prieto  
E. Nilsen-Nygaard  
A. Tripathy  
Cisco Systems  
February 23, 2017

Subscribing to Event Notifications  
draft-ietf-netconf-subscribed-notifications-00

Abstract

This document defines capabilities and operations for subscribing to content and providing asynchronous notification message delivery on that content. Notification delivery can occur over a variety of protocols used commonly in conjunction with YANG, such as NETCONF and RESTCONF. The capabilities and operations defined in this document when using in conjunction with draft-ietf-netconf-netconf-event-notifications are intended to obsolete RFC 5277.

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 <http://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 August 27, 2017.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

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

## Table of Contents

1. Introduction . . . . .	3
1.1. Motivation . . . . .	3
1.2. Terminology . . . . .	4
1.3. Solution Overview . . . . .	5
2. Solution . . . . .	6
2.1. Event Streams . . . . .	6
2.2. Filters . . . . .	6
2.3. Subscription State Model at the Publisher . . . . .	6
3. Data Model Trees for Event Notifications . . . . .	7
4. Dynamic Subscriptions . . . . .	11
4.1. Establishing a Subscription . . . . .	11
4.2. Modifying a Subscription . . . . .	12
4.3. Deleting a Subscription . . . . .	12
4.4. Killing a Subscription . . . . .	13
5. Configured Subscriptions . . . . .	13
5.1. Establishing a Configured Subscription . . . . .	14
5.2. Modifying a Configured Subscription . . . . .	16
5.3. Deleting a Configured Subscription . . . . .	16
6. Event (Data Plane) Notifications . . . . .	17
7. Subscription State Notifications . . . . .	18
7.1. subscription-started . . . . .	18
7.2. subscription-modified . . . . .	18
7.3. subscription-terminated . . . . .	19
7.4. subscription-suspended . . . . .	19
7.5. subscription-resumed . . . . .	19
7.6. notification-complete . . . . .	19
7.7. replay-complete . . . . .	19
8. Administrative Functions . . . . .	20
8.1. Subscription Monitoring . . . . .	20
8.2. Capability Advertisement . . . . .	20
8.3. Event Stream Discovery . . . . .	21
9. Data Model for Event Notifications . . . . .	21
10. Considerations . . . . .	41
10.1. Implementation Considerations . . . . .	41
10.2. Security Considerations . . . . .	41
11. Acknowledgments . . . . .	42
12. References . . . . .	42
12.1. Normative References . . . . .	42

12.2. Informative References . . . . .	43
Appendix A. Issues that are currently being worked and resolved	43
Appendix B. Changes between revisions . . . . .	44
Authors' Addresses . . . . .	45

## 1. Introduction

This document defines mechanisms that provide an asynchronous message notification delivery service in a protocol-agnostic manner. This document defines capabilities and operations for providing asynchronous message notification delivery for notifications including those necessary to establish, monitor, and support subscriptions to notification delivery.

Notification delivery can occur over a variety of protocols used commonly in conjunction with YANG, such as NETCONF [RFC6241] (defined in [I-D.ietf-netconf-netconf-event-notif]) and Restconf [RFC8040] (defined in [I-D.ietf-netconf-restconf-notif]). The capabilities and operations defined in this document are intended to obsolete RFC 5277, along with their mapping onto NETCONF transport.

### 1.1. Motivation

The motivation for this work is to enable the sending of transport agnostic asynchronous notification messages driven by a YANG Subscription that are consistent with the data model (content) and security model. Predating this work was [RFC5277] which defined a limited defines a notification mechanism for for NETCONF. However, there are various [RFC5277] has limitations, many of which have been exposed in [RFC7923].

The scope of the work aims at meeting the operational needs of network subscriptions:

- o Ability to dynamically or statically subscribe to event notifications available on a publisher.
- o Ability to negotiate acceptable dynamic subscription parameters.
- o Ability to filter the subset of notifications to be pushed with stream-specific semantics.
- o Ability for the notification payload to be interpreted independently of the transport protocol. (In other words, the encoded notification fully describes itself.)
- o Mechanism to communicate the notifications.

- o Ability to replay locally logged notifications.

## 1.2. 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].

**Configured subscription:** A subscription installed via a configuration interface which persists across reboots.

**Dynamic subscription:** A subscription agreed between subscriber and publisher created via RPC subscription state signaling messages.

**Event:** An occurrence of something that may be of interest. (e.g., a configuration change, a fault, a change in status, crossing a threshold, or an external input to the system.)

**Event notification:** A set of information intended for a Receiver indicating that one or more Event(s) have occurred. Details of the Event(s) may be included within the Notification.

**Filter:** Evaluation criteria, which may be applied against a targeted set of objects/events in a subscription. Information traverses the filter only if specified filter criteria are met.

**NACM:** NETCONF Access Control Model.

**OAM:** Operations, Administration, Maintenance.

**Publisher:** An entity responsible for streaming event notifications per the terms of a Subscriptions

**Receiver:** A target to which a publisher pushes event notifications. For dynamic subscriptions, the receiver and subscriber will often be the same entity.

**RPC:** Remote Procedure Call.

**Stream (also referred to as "event stream"):** A continuous ordered set of events grouped under an explicit criteria.

**Subscriber:** An entity able to request and negotiate a contract for the receipt of event notifications from a publisher.

**Subscription:** A contract with a publisher, stipulating which information receiver(s) wishes to have pushed from the publisher without the need for further solicitation.

### 1.3. Solution Overview

This document describes mechanisms for subscribing and receiving event notifications from an event server publisher. This document has similarities to the capabilities originally defined in [RFC5277]. This document extends the supported capabilities, and generalizes functionality to be protocol-agnostic.

Some enhancements over [RFC5277] include the ability to have multiple subscriptions on a single transport session, to terminate a single subscriptions without terminating the transport session, and to modify existing subscriptions.

The solution supports subscribing to event notifications using two mechanisms:

1. Dynamic subscriptions, where a subscriber initiates a subscription negotiation with a publisher via RPC. If the publisher wants to serve this request, it will accept it, and then start pushing event notifications. If the publisher does not wish to serve it as requested, then an error response is returned. This response may include hints at subscription parameters which would have been accepted.
2. Configured subscriptions, which is an optional mechanism that enables managing subscriptions via a configuration interface so that a publisher can send event notifications to configured receiver(s).

Some key characteristics of configured and dynamic subscriptions include:

- o The lifetime of a dynamic subscription is limited by the lifetime of the subscriber session used to establish it. Typically loss of the transport session tears down any dependent dynamic subscriptions.
- o The lifetime of a configured subscription is driven by configuration being present on the running configuration. This implies configured subscriptions persist across reboots, and persists even when transport is unavailable.
- o Subscriptions can be modified or terminated at any point of their lifetime. Configured subscriptions can be modified by any configuration client with write rights on the configuration of the subscription.

Note that there is no mixing-and-matching of dynamic and configured subscriptions. Specifically, a configured subscription cannot be modified or deleted using RPC. Similarly, a subscription established via RPC cannot be modified through configuration operations.

The publisher may decide to terminate a dynamic subscription at any time. Similarly, it may decide to temporarily suspend the sending of event notifications for either configured or dynamic subscriptions. Such termination or suspension may be driven by the publisher running out of resources to serve the subscription, or by internal errors on the publisher.

## 2. Solution

### 2.1. Event Streams

An event stream is a set of events available for subscription from a publisher. It is out of the scope of this document to identify a) how streams are defined, b) how events are defined/generated, and c) how events are assigned to streams.

That said, there is one standardized event stream, this is the "NETCONF" event stream. The NETCONF event stream contains all NETCONF XML event information supported by the publisher, except for where it has been explicitly indicated that this info must be excluded from the NETCONF stream.

As events are raised by a system, they may be assigned to one or more streams. The event is distributed to receivers meeting all three criteria: (1) a subscription includes the identified stream, (2) subscription filtering allows the event to traverse, and (3) no access control rules prohibit the receiver from receiving the event.

### 2.2. Filters

a publisher implementation SHOULD support the ability to perform filtering of notification records per [RFC5277]. (TODO: since 5277 is to be obsoleted, we should describe the filter here.)

### 2.3. Subscription State Model at the Publisher

Below is the state machine of a subscription for the publisher. It is important to note that a subscription doesn't exist at the publisher until it is accepted and made active. The mere request by a subscriber to establish a subscription is insufficient for that asserted subscription to be externally visible via this state machine.

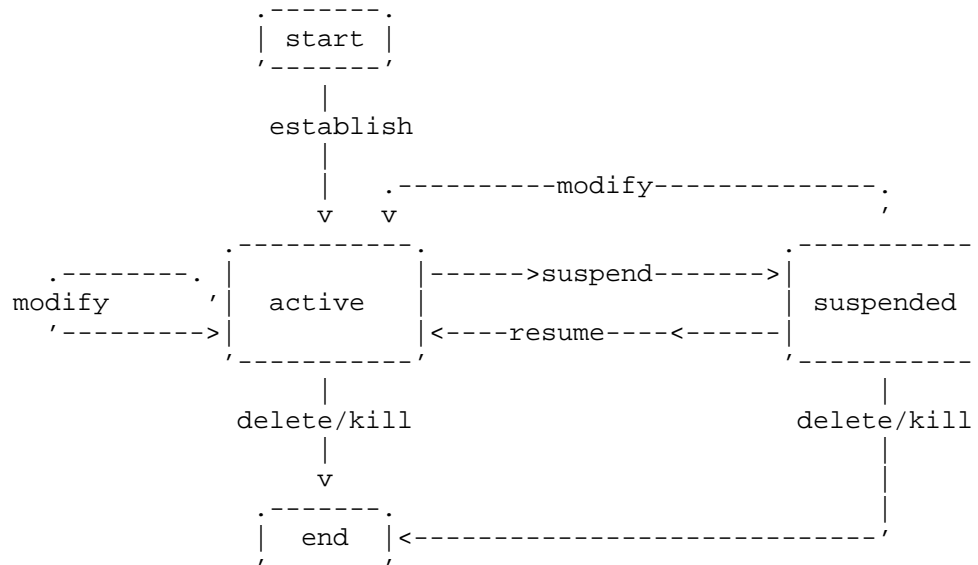


Figure 1: Subscription states at publisher

Of interest in this state machine are the following:

- o Successful <establish-subscription> or <modify-subscription> requests put the subscription into an active state.
- o Failed <modify-subscription> requests will leave the subscription in its previous state, with no visible change to any streaming updates.
- o A <delete-subscription> or <kill-subscription> will end the subscription.

### 3. Data Model Trees for Event Notifications

The YANG data model for event notifications is depicted in this section.

```

module: ietf-subscribed-notifications
  +--ro streams
  |   +--ro stream*   stream
  +--rw filters
  |   +--rw filter* [identifier]
  |   |   +--rw identifier   filter-id
  |   |   +--rw (filter-type)?
  |   |   |   +--:(by-reference)
  
```



```

|         |  +--rw filter-ref?   filter-ref
|         |  +---:(event-filter)
|         |  +--rw filter?
+--rw subscription-config {configured-subscriptions}?
|  +--rw subscription* [identifier]
|  |  +--rw identifier           subscription-id
|  |  +--rw stream?             stream
|  |  +--rw encoding?           encoding
|  |  +--rw stop-time?          yang:date-and-time
|  |  +--rw (filter-type)?
|  |  |  +---:(by-reference)
|  |  |  |  +--rw filter-ref?       filter-ref
|  |  |  |  +---:(event-filter)
|  |  |  |  +--rw filter?
|  |  +--rw receivers
|  |  |  +--rw receiver* [address port]
|  |  |  |  +--rw address           inet:host
|  |  |  |  +--rw port             inet:port-number
|  |  |  |  +--rw protocol?        transport-protocol
|  |  +--rw (notification-origin)?
|  |  |  +---:(interface-originated)
|  |  |  |  +--rw source-interface? if:interface-ref
|  |  |  +---:(address-originated)
|  |  |  |  +--rw source-vrf?       string
|  |  |  |  +--rw source-address?  inet:ip-address-no-zone
+--ro subscriptions
|  +--ro subscription* [identifier]
|  |  +--ro identifier           subscription-id
|  |  +--ro configured-subscription?
|  |  |  empty {configured-subscriptions}?
|  |  +--ro stream?             stream
|  |  +--ro encoding?           encoding
|  |  +--ro replay-start-time?   yang:date-and-time
|  |  +--ro stop-time?          yang:date-and-time
|  |  +--ro (filter-type)?
|  |  |  +---:(by-reference)
|  |  |  |  +--ro filter-ref?       filter-ref
|  |  |  |  +---:(event-filter)
|  |  |  |  +--ro filter?
|  |  +--ro (notification-origin)?
|  |  |  +---:(interface-originated)
|  |  |  |  +--ro source-interface? if:interface-ref
|  |  |  +---:(address-originated)
|  |  |  |  +--ro source-vrf?       string
|  |  |  |  +--ro source-address?  inet:ip-address-no-zone
+--ro receivers
|  |  +--ro receiver* [address]
|  |  |  +--ro address             inet:host

```

```

|      +---ro port                inet:port-number
|      +---ro protocol?           transport-protocol
|      +---ro pushed-notifications? yang:counter64
|      +---ro excluded-notifications? yang:counter64
+---ro subscription-status?        subscription-status

rpcs:
+---x establish-subscription
| +---w input
| | +---w stream?                stream
| | +---w encoding?              encoding
| | +---w replay-start-time?     yang:date-and-time
| | +---w stop-time?             yang:date-and-time
| | +---w (filter-type)?
| | | +---:(by-reference)
| | | | +---w filter-ref?        filter-ref
| | | +---:(event-filter)
| | | +---w filter?
| +---ro output
| | +---ro subscription-result    subscription-result
| | +---ro (result)?
| | | +---:(no-success)
| | | | +---ro filter-failure?    string
| | | | +---ro replay-start-time-hint? yang:date-and-time
| | | +---:(success)
| | +---ro identifier            subscription-id
+---x modify-subscription
| +---w input
| | +---w identifier?            subscription-id
| | +---w stop-time?            yang:date-and-time
| | +---w (filter-type)?
| | | +---:(by-reference)
| | | | +---w filter-ref?        filter-ref
| | | +---:(event-filter)
| | | +---w filter?
| +---ro output
| | +---ro subscription-result    subscription-result
| | +---ro (result)?
| | | +---:(no-success)
| | | | +---ro filter-failure?    string
+---x delete-subscription
| +---w input
| | +---w identifier            subscription-id
| +---ro output
| | +---ro subscription-result    subscription-result
+---x kill-subscription
| +---w input
| | +---w identifier            subscription-id

```

```

    +--ro output
      +--ro subscription-result    subscription-result

notifications:
+---n replay-complete
|   +--ro identifier    subscription-id
+---n notification-complete
|   +--ro identifier    subscription-id
+---n subscription-started
|   +--ro identifier    subscription-id
|   +--ro stream?      stream
|   +--ro encoding?    encoding
|   +--ro replay-start-time? yang:date-and-time
|   +--ro stop-time?   yang:date-and-time
|   +--ro (filter-type)?
|       +--:(by-reference)
|           |   +--ro filter-ref?    filter-ref
|           +--:(event-filter)
|               +--ro filter?
+---n subscription-resumed
|   +--ro identifier    subscription-id
+---n subscription-modified
|   +--ro identifier    subscription-id
|   +--ro stream?      stream
|   +--ro encoding?    encoding
|   +--ro replay-start-time? yang:date-and-time
|   +--ro stop-time?   yang:date-and-time
|   +--ro (filter-type)?
|       +--:(by-reference)
|           |   +--ro filter-ref?    filter-ref
|           +--:(event-filter)
|               +--ro filter?
+---n subscription-terminated
|   +--ro identifier    subscription-id
|   +--ro error-id      subscription-errors
|   +--ro filter-failure? string
+---n subscription-suspended
|   +--ro identifier    subscription-id
|   +--ro error-id      subscription-errors
|   +--ro filter-failure? string

```

The data model is structured as follows:

- o "Streams" contains a list of event streams that are supported by the publisher and that can be subscribed to.
- o "Filters" contains a configurable list of filters that can be applied to a subscription. This allows users to reference an

existing filter definition as an alternative to defining a filter inline for each subscription.

- o "Subscription-config" contains the configuration of configured subscriptions. The parameters of each configured subscription are a superset of the parameters of a dynamic subscription and use the same groupings. In addition, the configured subscriptions must also specify intended receivers and may specify the push source from which to send the stream of notification messages.
- o "Subscriptions" contains a list of all subscriptions on a publisher, both configured and dynamic. It can be used to retrieve information about the subscriptions which an publisher is serving.

The data model also contains a number of notifications that allow a publisher to signal information about a subscription. Finally, the data model contains a number of RPC definitions that are used to manage dynamic subscriptions.

#### 4. Dynamic Subscriptions

Dynamic subscriptions are managed via RPC.

##### 4.1. Establishing a Subscription

The <establish-subscription> operation allows a subscriber to request the creation of a subscription via RPC.

The input parameters of the operation are:

- o A filter which identifies what is being subscribed to, as well as what should be included (or not) in the pushed results.
- o An optional stream which may identify or reduce the domain of events against which the subscription is applied.
- o The desired encoding for the returned events. By default, updates are encoded using XML. Other encodings may be supported, such as JSON.
- o An optional stop time for the subscription.
- o An optional start time which indicates that this subscription is requesting a replay push of events previously generated.

If the publisher cannot satisfy the <establish-subscription> request, it sends a negative <subscription-result> element. If the subscriber

has no authorization to establish the subscription, the <subscription-result> indicates an authorization error. Optionally, the <subscription-result> may include one or more hints on alternative input parameters and value which would have resulted in an accepted subscription.

Subscription requests must fail if a filter with invalid syntax is provided or if the name of a non-existent stream is provided.

#### 4.1.1. Replay Subscription

The presence of a start time indicates that this is a replay subscription. The start time must be earlier than the current time. If the start time points earlier than the maintained history of Publisher's event buffer, then the subscription must be rejected. In this case the error response to the <establish-subscription> request should include a start time supportable by the Publisher.

#### 4.2. Modifying a Subscription

The <modify-subscription> operation permits changing the terms of an existing dynamic subscription previously established on that transport session. Subscriptions created by configuration operations cannot be modified via this RPC. Dynamic subscriptions can be modified one or multiple times. If the publisher accepts the requested modifications, it immediately starts sending events based on the new terms, completely ignoring the previous ones. If the publisher rejects the request, the subscription remains as prior to the request. That is, the request has no impact whatsoever. The contents of a such a rejected modification may include one or more hints on alternative input parameters and value which would have resulted in a successfully modified subscription.

Dynamic subscriptions established via RPC can only be modified (or deleted) via RPC using the same transport session used to establish that subscription.

#### 4.3. Deleting a Subscription

The <delete-subscription> operation permits canceling an existing subscription previously established on that transport session. If the publisher accepts the request, it immediately stops sending events for the subscription. If the publisher rejects the request, all subscriptions remain as prior to the request. That is, the request has no impact whatsoever.

Subscriptions established via RPC can only be deleted via RPC using the same transport session used for subscription establishment.

Configured subscriptions cannot be deleted using RPCs. Instead, configured subscriptions are deleted as part of regular configuration operations. Publishers MUST reject any RPC attempt to delete configured subscriptions.

#### 4.4. Killing a Subscription

The <kill-subscription> operation permits an operator to end any dynamic subscription. The publisher must accept the request for any dynamic subscription, and immediately stop sending events.

Configured subscriptions cannot be kill 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.

### 5. Configured Subscriptions

A configured subscription is a subscription installed via a configuration interface.

Configured subscriptions persist across reboots, and persist even when transport is unavailable.

Configured subscriptions can be modified by any configuration client with write permissions for the configuration of the subscription. Subscriptions can be modified or terminated via the configuration interface at any point of their lifetime.

Supporting configured subscriptions is optional and advertised using the "configured-subscriptions" feature.

In addition to subscription parameters that apply to dynamic subscriptions, the following additional parameters apply to configured subscriptions:

- o One or more receiver IP addresses (and corresponding ports)intended as the destination for push updates for each subscription. In addition, the transport protocol for each destination may be defined.
- o Optional parameters to identify an egress interface or IP address / VRF where a subscription updates should be pushed from the publisher. If not included, push updates will go off a default interface for the device.

### 5.1. Establishing a Configured Subscription

Configured subscriptions are established using configuration operations against the top-level subtree subscription-config. There are two key differences between RPC and <edit-config> RPC operations for subscription establishment. Firstly, <edit-config> operations install a subscription without question, while RPCs may support negotiation and rejection of requests. Secondly, while RPCs mandate that the subscriber establishing the subscription is the only receiver of the notifications, <edit-config> operations permit specifying receivers independent of any tracked subscriber. Immediately after a subscription is successfully established, the publisher sends to any newly active receivers a control-plane notification stating the subscription has been established (subscription-started).

Because there is no explicit association with an existing transport session, <edit-config> operations require additional parameters to indicate the receivers of the notifications and possibly the source of the notifications such as a specific egress interface.

For example at subscription establishment if NETCONF transport is being used, a client may send:

```
<rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
  xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <subscription-config
      xmlns="urn:ietf:params:xml:ns:netconf:notification:2.0">
      <subscription>
        <subscription-id>
          1922
        </subscription-id>
        <stream>
          foo
        </stream>
        <receiver>
          <address>
            1.2.3.4
          </address>
          <port>
            1234
          </port>
        </receiver>
      </subscription>
    </subscription-config>
  </edit-config>
</rpc>
```

Figure 2: Configured subscription creation via NETCONF

if the request is accepted, the publisher would reply:

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

Figure 3: Successful NETCONF configured subscription response

if the request is not accepted because the publisher cannot serve it, the publisher may reply:



```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <rpc-error>
    <error-type>application</error-type>
    <error-tag>resource-denied</error-tag>
    <error-severity>error</error-severity>
    <error-message xml:lang="en">
      Temporarily the publisher cannot serve this
      subscription due to the current workload.
    </error-message>
  </rpc-error>
</rpc-reply>
```

Figure 4: A NETCONF response for a failed configured subscription creation

## 5.2. Modifying a Configured Subscription

Configured subscriptions can be modified using configuration operations against the top-level subtree subscription-config.

Immediately after a subscription is successfully modified, the publisher sends to the existing receivers a control-plane notification stating the subscription has been modified (i.e., subscription-modified).

If the modification involved adding and/or removing receivers, those modified receivers are sent control-plane notifications, indicating they have been added (i.e., subscription-started to a specific receiver) or removed (i.e., subscription-terminated to a specific receiver.)

## 5.3. Deleting a Configured Subscription

Subscriptions can be deleted using configuration operations against the top-level subtree subscription-config. For example, in RESTCONF:

```
DELETE /subscription-config/subscription=1922 HTTP/1.1
Host: example.com

HTTP/1.1 204 No Content
Date: Sun, 24 Jul 2016 11:23:40 GMT
Server: example-server
```

Figure 5: Deleting a configured subscription

Immediately after a subscription is successfully deleted, the publisher sends to all receivers of that subscription a control-plane

notification stating the subscription has been terminated (subscription-terminated).

## 6. Event (Data Plane) Notifications

Once a subscription has been set up, the publisher streams (asynchronously) notifications per the terms of the subscription. We refer to these as event notifications. For dynamic subscriptions set up via RPC operations, event notifications are sent over the session used to establish the subscription. For configured subscriptions, event notifications are sent over the specified connections.

An event notification is sent to a receiver when something of interest occurs which is able to traverse all specified filtering and access control criteria. The event notification must include:

- o a subscription-id element of type uint32 which corresponds to responsible subscription in the Publisher.
- o an eventTime element which provides the time the event was generated by the event source. This event time parameter is of type dateTime and compliant to [RFC3339]. Implementations must support time zones.
- o the event notification content tagged and provided by a source in the publisher.

The following is an example of a compliant event notification. This example extending the example within [RFC7950] section 7.16.3 to include the mandatory information described above:

```
<notification
  xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2007-09-01T10:00:00Z</eventTime>
  <subscription-id>500</subscription-id>
  <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 6: Data plane notification

While this extended [RFC7950] section 7.16 notification provides a valid method of encapsulating subscribed notifications, other transport encapsulation methods are also viable. Improvements may be achieved in some implementations in the following ways:

- o transport efficiency may be gained by allowing the encapsulation and bundled push of multiple events within the same event notification.
- o identifiers to designate the current and previous event notification can be used to discover duplicated and dropped notifications
- o additional header types can be used to pass relevant metadata.
- o a signature or hash can be included to verify the efficacy of the Publisher

This is being explored in NETMOD Notifications 2.0 [I-D.voit-notifications2].

## 7. Subscription State Notifications

In addition to data plane notifications, a publisher may send subscription state notifications to indicate to receivers that an event related to the subscription management has occurred.

Subscription state notifications are unlike other notifications in that they are not general-purpose notifications. They cannot be filtered out, and they are delivered only to directly impacted receiver(s) of a subscription. The definition of subscription state notifications is distinct from other notifications by making use of a YANG extension tagging them as subscription state notification.

Subscription state notifications include indications that a replay of notifications has been completed, that a subscription is done sending notifications because an end time has been reached, and that a subscription has started, been modified, been terminated, or been suspended. They are described in the following subsections.

### 7.1. subscription-started

This notification indicates that a configured subscription has started and data updates are beginning to be sent. This notification includes the parameters of the subscription, except for the receiver(s) addressing information and push-source information. Note that for RPC-based subscriptions, no such notifications are sent.

### 7.2. subscription-modified

This notification indicates that a configured subscription has been modified successfully. This notification includes the parameters of the subscription, except for the receiver(s) addressing information

and push-source information. Note that for RPC-based subscriptions, no such notifications are sent.

#### 7.3. subscription-terminated

This notification indicates that a subscription has been terminated by the publisher. The notification includes the reason for the termination. The publisher may decide to terminate a subscription when it is running out of resources for serving it, an internal error occurs, etc. Publisher-driven terminations are notified to all receivers. The management plane can also terminate configured subscriptions using configuration operations.

Subscribers can terminate via RPC subscriptions established via a delete-subscription RPC. In such cases, no subscription-terminated notifications are sent. However if a kill-subscription RPC is sent, or some other event results in the end of a subscription, then there must be a notification that the subscription has been ended.

#### 7.4. subscription-suspended

This notification indicates that a publisher has suspended a subscription. The notification includes the reason for the suspension. A possible reason is the lack of resources to serve it. No further data plane notifications will be sent until the subscription resumes. Suspensions are notified to the subscriber (in the case of dynamic subscriptions) and all receivers (in the case of configured subscriptions).

#### 7.5. subscription-resumed

This notification indicates that a previously suspended subscription has been resumed. Data plane notifications generated in the future will be sent after the subscription terms. Resumptions are notified to the subscriber (in the case of dynamic subscriptions) and all receivers (in the case of configured subscriptions).

#### 7.6. notification-complete

This notification is sent to indicate that a subscription, which includes a stop time, has finished passing events.

#### 7.7. replay-complete

This notification indicates that all of the notifications prior to the current time have been sent. This includes new notifications generated since the start of the subscription. This notification must not be sent for any other reason.

If subscription contains no stop time, or has a stop time which has not been reached, then after the replay-complete notification has been sent notifications will be sent in sequence as they arise naturally within the system.

## 8. Administrative Functions

### 8.1. Subscription Monitoring

Container "subscriptions" in the YANG module below contains the state of all subscriptions that are currently active. This includes subscriptions that were established (and have not yet been deleted) using RPCs, as well as subscriptions that have been configured as part of configuration. Using the <get> operation with NETCONF, or subscribing to this information via [I-D.ietf-netconf-yang-push] allows the status of subscriptions to be monitored.

Each subscription is represented as a list element. The associated information includes an identifier for the subscription, a subscription status, as well as the various subscription parameters that are in effect. The subscription status indicates whether the subscription is currently active and healthy, or if it is degraded in some form. Leaf "configured-subscription" indicates whether the subscription came into being via configuration or via RPC.

Subscriptions that were established by RPC are removed from the list once they expire (reaching stop-time) or when they are terminated. Subscriptions that were established by configuration need to be deleted from the configuration by a configuration editing operation even if the stop time has been passed.

### 8.2. Capability Advertisement

Capabilities are advertised in messages sent by each peer during session establishment [RFC6241]. Publishers supporting the features in this document must advertise the capability "urn:ietf:params:netconf:capability:notification:2.0".

The mechanism defined in this document is identified by "urn:ietf:params:netconf:capability:notification:2.0". If a subscriber only supports [RFC5277] and not this specification, then they will recognize the capability "urn:ietf:params:netconf:capability:notification:1.0" and ignore the capability defined in this document.

### 8.3. Event Stream Discovery

A publisher maintains a list of available event streams as operational data. This list contains both standardized and vendor-specific event streams. A client can retrieve this list like any other YANG-defined data, for example using the <get> operation when using NETCONF.

## 9. Data Model for Event Notifications

```
<CODE BEGINS> file "ietf-subscribed-notifications.yang"
module ietf-subscribed-notifications {
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications";

  prefix sn;

  import ietf-yang-types {
    prefix yang;
  }
  import ietf-inet-types {
    prefix inet;
  }
  import ietf-interfaces {
    prefix if;
  }

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

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

    WG Chair: Mehmet Ersue
               <mailto:mehmet.ersue@nokia.com>

    Editor:      Alexander Clemm
               <mailto:ludwig@clemm.org>

    Editor:      Eric Voit
               <mailto:evoit@cisco.com>

    Editor:      Alberto Gonzalez Prieto
               <mailto:albertgo@cisco.com>

    Editor:      Einar Nilsen-Nygaard
```

<mailto:einarinn@cisco.com>;

Editor: Ambika Prasad Tripathy  
<mailto:ambtripa@cisco.com>;

```
description
  "This module contains conceptual YANG specifications for NETCONF
  Event Notifications.";

revision 2017-02-23 {
  description
    "Tweaks to remove two notifications, RPC for create subscription
    refined with stream default, new grouping to eliminate some
    dynamically modifiable parameters in modify subscription RPC";
  reference
    "draft-ietf-netconf-subscribed-notifications-00";
}

/*
 * FEATURES
 */

feature json {
  description
    "This feature indicates that JSON encoding of notifications
    is supported.";
}

feature configured-subscriptions {
  description
    "This feature indicates that management plane configuration
    of subscription is supported.";
}

/*
 * EXTENSIONS
 */

extension subscription-state-notif {
  description
    "This statement applies only to notifications. It indicates that
    the notification is a subscription state notification (aka OAM
    notification). Therefore it does not participate in a regular
    event stream and does not need to be specifically subscribed
    in order to receive notifications.";
}
```

```
/*
 * IDENTITIES
 */

/* Identities for streams */
identity stream {
  description
    "Base identity to represent a generic stream of event
    notifications.";
}

identity NETCONF {
  base stream;
  description
    "Default NETCONF event stream, containing events based on
    notifications defined as YANG modules that are supported by the
    system. This contains the same set of events in a default
    RFC-5277 NETCONF stream";
}

/* Identities for subscription results */
identity subscription-result {
  description
    "Base identity for RPC responses to requests surrounding
    management (e.g. creation, modification, deletion) of
    subscriptions.";
}

identity ok {
  base subscription-result;
  description
    "OK - RPC was successful and was performed as requested.";
}

identity error {
  base subscription-result;
  description
    "RPC was not successful.
    Base identity for error return codes.";
}

/* Identities for subscription stream status */
identity subscription-stream-status {
  description
    "Base identity for the status of subscriptions and datastreams.";
}

identity active {
```



```
    base subscription-stream-status;
    description
        "Status is active and healthy.";
}

identity inactive {
    base subscription-stream-status;
    description
        "Status is inactive, for example outside the interval between
        start time and stop time.";
}

identity suspended {
    base subscription-stream-status;
    description
        "The status is suspended, meaning that the publisher is currently
        unable to provide the negotiated updates for the subscription.";
}

identity in-error {
    base subscription-stream-status;
    description
        "The status is in error or degraded, meaning that stream and/or
        subscription is currently unable to provide the negotiated
        notifications.";
}

/* Identities for subscription errors */

identity internal-error {
    base error;
    description
        "Error within publisher prohibits operation.";
}

identity suspension-timeout {
    base error;
    description
        "Termination of previously suspended subscription. The publisher
        has eliminated the subscription as it exceeded a time limit for
        suspension.";
}

identity stream-unavailable {
    base error;
    description
        "Stream name does not exist or is not available to the receiver.";
}
```

```
identity encoding-unavailable {
  base error;
  description
    "Encoding not supported";
}

identity replay-unsupported {
  base error;
  description
    "Replay cannot be performed for this subscription. The publisher
    does not provide the requested historic information via replay.";
}

identity history-unavailable {
  base error;
  description
    "Replay request too far into the past. The publisher does store
    historic information for all parts of requested subscription, but
    not back to the requested timestamp.";
}

identity filter-unavailable {
  base error;
  description
    "Referenced filter does not exist";
}

identity filter-unsupported {
  base error;
  description
    "Cannot parse syntax within the filter. Failure can be from a
    syntax error, or a syntax too complex to be processed by the
    platform. The supplemental info should include the invalid part
    of the filter.";
}

identity namespace-unavailable {
  base error;
  description
    "Referenced namespace doesn't exist or is unavailable
    to the receiver.";
}

identity no-such-subscription {
  base 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 acceptable subscription which has been
        statically configured.";
    }

    /* Identities for encodings */
    identity encodings {
        description
            "Base identity to represent data encodings";
    }

    identity encode-xml {
        base encodings;
        description
            "Encode data using XML";
    }

    identity encode-json {
        base encodings;
        description
            "Encode data using JSON";
    }

    /* Identities for transports */
    identity transport {
        description
            "An identity that represents a transport protocol for event
            notifications";
    }

    identity netconf {
        base transport;
        description
            "Netconf notifications as a transport.";
    }

    /*
     * TYPEDEFS
     */

    typedef subscription-id {
        type uint32;
        description
            "A type for subscription identifiers.";
    }

    typedef filter-id {
        type uint32;
        description
```

```
    "A type to identify filters which can be associated with a
      subscription.";
  }

  typedef subscription-result {
    type identityref {
      base subscription-result;
    }
    description
      "The result of a subscription operation";
  }

  typedef subscription-errors {
    type identityref {
      base error;
    }
    description
      "The reason for the failure of an RPC request or the sending of a
        subscription suspension or termination notification";
  }

  typedef encoding {
    type identityref {
      base encodings;
    }
    description
      "Specifies a data encoding, e.g. for a data subscription.";
  }

  typedef subscription-status {
    type identityref {
      base subscription-stream-status;
    }
    description
      "Specifies the status of a subscription or datastream.";
  }

  typedef transport-protocol {
    type identityref {
      base transport;
    }
    description
      "Specifies transport protocol used to send notifications to a
        receiver.";
  }

  typedef notification-origin {
    type enumeration {
```

```
    enum "interface-originated" {
        description
            "Notifications will be sent from a specific interface on a
            publisher";
    }
    enum "address-originated" {
        description
            "Notifications will be sent from a specific address on a
            publisher";
    }
}
description
    "Specifies from where notifications will be sourced when
    being sent by the publisher.";
}

typedef stream {
    type identityref {
        base stream;
    }
    description
        "Specifies a system-provided datastream.";
}

typedef filter-ref {
    type leafref {
        path "/sn:filters/sn:filter/sn:identifier";
    }
    description
        "This type is used to reference a filter.";
}

/*
 * GROUPINGS
 */

grouping base-filter {
    description
        "This grouping defines the base for filters for notification
        events.";
    choice filter-type {
        description
            "A filter needs to be a single filter of a given type.  Mixing
            and matching of multiple filters does not occur at the level of
            this grouping.";
        case by-reference {
            description
                "Incorporate a filter that has been configured separately.";
        }
    }
}
```

```
    leaf filter-ref {
      type filter-ref;
      description
        "References an existing filter which is to be applied to
         the potential events of the subscription.";
    }
  }
  case event-filter {
    anyxml filter {
      description
        "Filter which excludes whole event-notifications. If a filter
         element is specified to look for data of a particular
         value, and the data item is not present within a particular
         event notification for its value to be checked against, the
         notification will be filtered out. For example, if one
         were to check for 'severity=critical' in a configuration
         event notification where this field was not supported, then
         the notification would be filtered out. For subtree
         filtering, a non-empty node set means that the filter
         matches. For XPath filtering, the mechanisms defined in
         [XPATH] should be used to convert the returned value to
         boolean.";
    }
  }
}

grouping subscription-policy-non-configurable {
  description
    "This grouping describes the information which can only be set
     in a dynamic subscription request via RPC.";
  leaf replay-start-time {
    type yang:date-and-time;
    description
      "Used to trigger the replay feature and indicate that the
       replay should start at the time specified. If replay-start-time
       is not present, this is not a replay subscription and event
       pushes should start immediately. It is never valid to
       specify start times that are later than or equal to the
       current time.";
  }
}

grouping subscription-policy-non-modifiable {
  description
    "This grouping describes the information in a subscription which
     should not change during the life of the subscription.";
  leaf stream {
```

```
    type stream;
    description
        "Indicates which stream of events is of interest.
        If not present, events in the default NETCONF stream
        will be sent.";
}
leaf encoding {
    type encoding;
    default "encode-xml";
    description
        "The type of encoding for the subscribed data.
        Default is XML";
}
}

grouping subscription-policy-modifiable {
    description
        "This grouping describes all objects which may be changed
        in a subscription via an RPC.";
    leaf stop-time {
        type yang:date-and-time;
        description
            "Identifies a time after which notification events should not
            be sent. If stop-time is not present, the notifications will
            continue until the subscription is terminated. If
            replay-start-time exists, stop-time must for a subsequent time.
            If replay-start-time doesn't exist, stop-time must for a future
            time.";
    }
    uses base-filter;
}

grouping subscription-policy {
    description
        "This grouping describes information concerning a subscription.";
    uses subscription-policy-non-modifiable;
    uses subscription-policy-non-configurable;
    uses subscription-policy-modifiable;
}

grouping notification-origin-info {
    description
        "Defines the sender source from which notifications for a
        configured subscription are sent.";
    choice notification-origin {
        description
            "Identifies the egress interface on the Publisher from which
            notifications will or are being sent.";
    }
}
```

```
case interface-originated {
  description
    "When the push source is out of an interface on the
    Publisher established via static configuration.";
  leaf source-interface {
    type if:interface-ref;
    description
      "References the interface for notifications.";
  }
}
case address-originated {
  description
    "When the push source is out of an IP address on the
    Publisher established via static configuration.";
  leaf source-vrf {
    type string;
    description
      "Network instance name for the VRF. This could also have
      been a leafref to draft-ietf-rtgwg-ni-model, but that model
      in not complete, and may not be implemented on a box.";
  }
  leaf source-address {
    type inet:ip-address-no-zone;
    description
      "The source address for the notifications.";
  }
}
}

grouping receiver-info {
  description
    "Defines where and how to get notifications for a configured
    subscriptions to one or more targeted recipient. This includes
    specifying the destination addressing as well as a transport
    protocol acceptable to the reciever.";
  container receivers {
    description
      "Set of receivers in a subscription.";
    list receiver {
      key "address port";
      min-elements 1;
      description
        "A single host or multipoint address intended as a target
        for the notifications for a subscription.";
      leaf address {
        type inet:host;
        mandatory true;
      }
    }
  }
}
```



```
        description
            "Specifies the address for the traffic to reach a remote
            host. One of the following must be specified: an ipv4
            address, an ipv6 address, or a host name.";
    }
    leaf port {
        type inet:port-number;
        mandatory true;
        description
            "This leaf specifies the port number to use for messages
            destined for a receiver.";
    }
    leaf protocol {
        type transport-protocol;
        default "netconf";
        description
            "This leaf specifies the transport protocol used
            to deliver messages destined for the receiver. Each
            protocol may use the address and port information
            differently as applicable.";
    }
}
}
}

grouping error-identifier {
    description
        "A code passed back within an RPC response to describe why the RFC
        has failed, or within a state change notification to describe why
        the change has occurred.";
    leaf error-id {
        type subscription-errors;
        mandatory true;
        description
            "Identifies the subscription error condition.";
    }
}

grouping error-hints {
    description
        "Objects passed back within an RPC response to describe why the RFC
        has failed, or within a state change notification to describe why
        the change has occurred.";
    leaf filter-failure {
        type string;
        description
            "Information describing where and/or why a provided filter was
            unsupportable for a subscription.";
    }
}
```

```
    }  
  }  
  
  grouping subscription-response-with-hints {  
    description  
      "Defines the output for the establish-subscription and  
      modify-subscription RPCs.";   
    leaf subscription-result {  
      type subscription-result;  
      mandatory true;  
      description  
        "Indicates whether subscription is operational, or if a problem  
        was encountered.";   
    }  
    choice result {  
      description  
        "Depending on the subscription result, different data is  
        returned.";   
      case no-success {  
        description  
          "This case applies when a subscription request was not  
          successful and no subscription was created (or modified) as a  
          result. In this case, information MAY be returned that  
          indicates suggested parameter settings that would have a  
          high likelihood of succeeding in a subsequent establish-  
          subscription or modify-subscription request.";   
        uses error-hints;  
      }  
    }  
  }  
}  
  
/*  
 * 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 (as indicated by  
    subscription-result) is returned, the subscription is not  
    created. In that case, the RPC output MAY include suggested  
    parameter settings that would have a high likelihood of  
    succeeding in a subsequent establish-subscription request.";   
  input {  

```

```
    uses subscription-policy;
  }
  output {
    uses subscription-response-with-hints {
      augment "result" {
        description
          "Allows information to be passed back as part of a
          successful subscription establishment.";
        case success {
          description
            "This case is used when the subscription request was
            successful.";
          leaf identifier {
            type subscription-id;
            mandatory true;
            description
              "Identifier used for this subscription.";
          }
        }
      }
    }
    augment "result/no-success" {
      description
        "Contains establish RPC specific objects which can be
        returned as hints for future attempts.";
      leaf replay-start-time-hint {
        type yang:date-and-time;
        description
          "If a replay has been requested, but the requested replay
          time cannot be honored, this may provide a hint at an
          alternate time which may be supportable.";
      }
    }
  }
}

rpc modify-subscription {
  description
    "This RPC allows a subscriber to modify a subscription that was
    previously created using establish-subscription.  If successful,
    the changed subscription remains in effect for the duration of
    the subscriber's association with the publisher, or until the
    subscription is again modified or terminated.  In case an error
    is returned (as indicated by subscription-result), the
    subscription is not modified and the original subscription
    parameters remain in effect.  In that case, the rpc error
    response MAY include suggested parameter hints that would have
    a high likelihood of succeeding in a subsequent
```

```
        modify-subscription request.";
    input {
        leaf identifier {
            type subscription-id;
            description
                "Identifier to use for this subscription.";
        }
        uses subscription-policy-modifiable;
    }
    output {
        uses subscription-response-with-hints;
    }
}

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.";
    input {
        leaf identifier {
            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.";
        }
    }
    output {
        leaf subscription-result {
            type subscription-result;
            mandatory true;
            description
                "Indicates whether subscription is operational, or if a
                problem was encountered.";
        }
    }
}

rpc kill-subscription {
    description
        "This RPC allows an operator to delete a dynamic subscription
        without restrictions on the originating subscriber or underlying
        transport session.";
    input {
        leaf identifier {
            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.";
    }
}
output {
    leaf subscription-result {
        type subscription-result;
        mandatory true;
        description
            "Indicates whether subscription is operational, or if a
             problem was encountered.";
    }
}
}

/*
 * NOTIFICATIONS
 */

notification replay-complete {
    sn:subscription-state-notif;
    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 identifier {
        type subscription-id;
        mandatory true;
        description
            "This references the affected subscription.";
    }
}

notification notification-complete {
    sn:subscription-state-notif;
    description
        "This notification is sent to indicate that a subscription, has
         finished passing events.";
    leaf identifier {
        type subscription-id;
        mandatory true;
        description
            "This references the affected subscription.";
    }
}
```

```
notification subscription-started {
  sn:subscription-state-notif;
  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 identifier {
    type subscription-id;
    mandatory true;
    description
      "This references the affected subscription.";
  }
  uses subscription-policy;
}

notification subscription-resumed {
  sn:subscription-state-notif;
  description
    "This notification indicates that a subscription that had
    previously been suspended has resumed. Notifications will once
    again be sent.";
  leaf identifier {
    type subscription-id;
    mandatory true;
    description
      "This references the affected subscription.";
  }
}

notification subscription-modified {
  sn:subscription-state-notif;
  description
    "This notification indicates that a subscription has been
    modified. Notifications sent from this point on will conform to
    the modified terms of the subscription.";
  leaf identifier {
    type subscription-id;
    mandatory true;
    description
      "This references the affected subscription.";
  }
  uses subscription-policy;
}

notification subscription-terminated {
  sn:subscription-state-notif;
  description
```

```
    "This notification indicates that a subscription has been
      terminated.";
  leaf identifier {
    type subscription-id;
    mandatory true;
    description
      "This references the affected subscription.";
  }
  uses error-identifier;
  uses error-hints;
}

notification subscription-suspended {
  sn:subscription-state-notif;
  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 identifier {
    type subscription-id;
    mandatory true;
    description
      "This references the affected subscription.";
  }
  uses error-identifier;
  uses error-hints;
}

/*
 * DATA NODES
 */

container streams {
  config false;
  description
    "This container contains a leaf list of built-in
      streams that are provided by the system.";
  leaf-list stream {
    type stream;
    description
      "Identifies the built-in streams that are supported by the
        system. Built-in streams are associated with their own
        identities, each of which carries a special semantics.
        In case configurable custom streams are supported,
        as indicated by the custom-stream identity, the configuration
```

```
        of those custom streams is provided separately.";
    }
}
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 filter {
        key "identifier";
        description
            "A list of configurable filters that can be applied to
            subscriptions.";
        leaf identifier {
            type filter-id;
            description
                "An identifier to differentiate between filters.";
        }
        uses base-filter;
    }
}
container subscription-config {
    if-feature "configured-subscriptions";
    description
        "Contains the list of subscriptions that are configured,
        as opposed to established via RPC or other means.";
    list subscription {
        key "identifier";
        description
            "Content of a subscription.";
        leaf identifier {
            type subscription-id;
            description
                "Identifier to use for this subscription.";
        }
        uses subscription-policy-non-modifiable;
        uses subscription-policy-modifiable;
        uses receiver-info {
            if-feature "configured-subscriptions";
        }
        uses notification-origin-info {
            if-feature "configured-subscriptions";
        }
    }
}
container subscriptions {
    config false;
    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 "identifier";
  config false;
  description
    "Content of a subscription.
     Subscriptions can be created using a control channel or RPC, or
     be established through configuration.";
  leaf identifier {
    type subscription-id;
    description
      "Identifier of this subscription.";
  }
  leaf configured-subscription {
    if-feature "configured-subscriptions";
    type empty;
    description
      "The presence of this leaf indicates that the subscription
       originated from configuration, not through a control channel
       or RPC.";
  }
  uses subscription-policy;
  uses notification-origin-info {
    if-feature "configured-subscriptions";
  }
  uses receiver-info {
    augment receivers/receiver {
      description
        "include operational data on configured receivers.";
      leaf pushed-notifications {
        type yang:counter64;
        description
          "Operational data which provides the number of update
           notifications pushed to a receiver.";
      }
      leaf excluded-notifications {
        type yang:counter64;
        description
          "Operational data which provides the number of non-
           datastore update notifications explicitly removed via
           filtering so that they are not sent to a receiver.";
      }
    }
  }
}
```

```
    leaf subscription-status {  
      type subscription-status;  
      description  
        "The status of the subscription.";  
    }  
  }  
}  
<CODE ENDS>
```

## 10. Considerations

### 10.1. Implementation Considerations

For a deployment including both configured and dynamic subscriptions, split subscription identifiers into static and dynamic halves. That way there should not be collisions if the configured subscriptions attempt to set a subscription-id which might have already been dynamically allocated.

The <notification> elements are never sent before the transport layer, including capabilities exchange, has been established.

### 10.2. Security Considerations

A secure transport is highly recommended and the publisher must ensure that the user has sufficient authorization to perform the function they are requesting against the specific subset of content involved. When a <get> is received that refers to the content defined in this memo, receivers should only be able to view the content for which they have sufficient privileges. <establish-subscription> operations can be considered like deferred <get>, and the content that different users can access may vary. This different access is reflected in the <notification> to which different users are able to subscribe.

The contents of notifications, as well as the names of event streams, may contain sensitive information and care should be taken to ensure that they are viewed only by authorized users. The publisher **MUST** NOT include any content in a notification that the user is not authorized to view.

If a malicious or buggy subscriber sends a number of <establish-subscription> requests, then these subscriptions accumulate and may use up system resources. In such a situation, subscriptions can be terminated by terminating the transport session. The publisher can also suspend or terminate subscriptions with per-subscription granularity.

A subscription could be configured on another receiver's behalf, with the goal of flooding that receiver with updates. One or more publishers could be used to overwhelm a receiver, which doesn't even support subscriptions. Subscribers that do not want pushed data need only terminate or refuse any transport sessions from the publisher. In addition, the NETCONF Authorization Control Model [RFC6536] SHOULD be used to control and restrict authorization of subscription configuration. This control models permits specifying per-user permissions to receive specific event notification types. The permissions are specified as a set of access control rules.

Note that streams can define additional authorization requirements. For instance, in [I-D.ietf-netconf-yang-push], each of the elements in its data plane notifications must also go through access control.

It is recommended that the NACM "very-secure" tag is placed on the <kill-subscription> RPC so that only administrators can access.

## 11. Acknowledgments

For their valuable comments, discussions, and feedback, we wish to acknowledge Andy Bierman, Tim Jenkins, Balazs Lengyel, Shaon Chisholm, Hector Trevino, Susan Hares, Kent Watsen, Michael Scharf, and Guangying Zheng.

## 12. References

### 12.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <<http://www.rfc-editor.org/info/rfc3339>>.
- [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, <<http://www.rfc-editor.org/info/rfc6241>>.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <<http://www.rfc-editor.org/info/rfc6536>>.

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

## 12.2. Informative References

- [I-D.ietf-netconf-netconf-event-notif]  
Clemm, Alexander., Voit, Eric., Gonzalez Prieto, Alberto., Nilsen-Nygaard, E., Tripathy, A., Chisholm, S., and H. Trevino, "NETCONF support for event notifications", August 2016, <<https://datatracker.ietf.org/doc/draft-ietf-netconf-netconf-event-notifications/>>.
- [I-D.ietf-netconf-restconf-notif]  
Voit, Eric., Clemm, Alexander., Tripathy, A., Nilsen-Nygaard, E., and Alberto. Gonzalez Prieto, "Restconf and HTTP transport for event notifications", August 2016, <<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., and E. Nilsen-Nygaard, "Subscribing to YANG datastore push updates", February 2017, <<https://datatracker.ietf.org/doc/draft-ietf-netconf-yang-push/>>.
- [I-D.voit-notifications2]  
Voit, Eric., Clemm, Alexander., Bierman, A., and T. Jenkins, "YANG Notification Headers and Bundles", February 2017, <<https://datatracker.ietf.org/draft-voit-netmod-yang-notifications2/>>.
- [RFC5277] Chisholm, S. and H. Trevino, "NETCONF Event Notifications", RFC 5277, DOI 10.17487/RFC5277, July 2008, <<http://www.rfc-editor.org/info/rfc5277>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<http://www.rfc-editor.org/info/rfc8040>>.

## Appendix A. Issues that are currently being worked and resolved

(To be removed by RFC editor prior to publication)

Issue #9: validate that Subscription ID will only be relevant locally to a single receiver

## Issue #6: Data plane notifications and layered headers

How to allow for seamless integration with non-standard encodings and transports (like GPB/GRPC). Specify requirements encoding and transport must meet, provide examples.

## Appendix B. Changes between revisions

(To be removed by RFC editor prior to publication)

## 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 as this is not RFC-5277 compatible.
- 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 Streams extracted in favor of more information in the filters section.
- o Operational counters per receiver.
- o Subscription-id and filter-id renamed to identifier
- o Section for replay added. Replay-start and stop-time updated. Replay now cannot be configured.
- o Control plane notification renamed to subscription state 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 changable via RPC. Removed notifications for adding and removing receivers of configured subscriptions.
- o Expanded/renamed defintions from event server to publisher, and client to subscriber as applicable. Updated the definitions to include and expand on RFC 5277.
- o Removal of redundant with other drafts
- o Many other clean-ups of wording and terminology

#### Authors' Addresses

Eric Voit  
Cisco Systems

Email: [evoit@cisco.com](mailto:evoit@cisco.com)

Alexander Clemm  
Huawei

Email: [ludwig@clemm.org](mailto:ludwig@clemm.org)

Alberto Gonzalez Prieto  
Cisco Systems

Email: [albertgo@cisco.com](mailto:albertgo@cisco.com)

Einar Nilsen-Nygaard  
Cisco Systems

Email: [einarnn@cisco.com](mailto:einarnn@cisco.com)

Ambika Prasad Tripathy  
Cisco Systems

Email: [ambtripa@cisco.com](mailto:ambtripa@cisco.com)

NETCONF  
Internet-Draft  
Intended status: Standards Track  
Expires: November 9, 2019

E. Voit  
Cisco Systems  
A. Clemm  
Huawei  
A. Gonzalez Prieto  
Microsoft  
E. Nilsen-Nygaard  
A. Tripathy  
Cisco Systems  
May 8, 2019

Subscription to YANG Event Notifications  
draft-ietf-netconf-subscribed-notifications-26

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

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 November 9, 2019.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

#### Table of Contents

1. Introduction . . . . .	3
1.1. Motivation . . . . .	3
1.2. Terminology . . . . .	4
1.3. Solution Overview . . . . .	5
1.4. Relationship to RFC 5277 . . . . .	6
2. Solution . . . . .	7
2.1. Event Streams . . . . .	7
2.2. Event Stream Filters . . . . .	8
2.3. QoS . . . . .	8
2.4. Dynamic Subscriptions . . . . .	9
2.5. Configured Subscriptions . . . . .	17
2.6. Event Record Delivery . . . . .	25
2.7. Subscription state change notifications . . . . .	26
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 . . . . .	63
5.3. Transport Requirements . . . . .	64
5.4. Security Considerations . . . . .	65
6. Acknowledgments . . . . .	68
7. References . . . . .	69



7.1. Normative References . . . . .	69
7.2. Informative References . . . . .	70
Appendix A. Example Configured Transport Augmentation . . . . .	71
Appendix B. Changes between revisions . . . . .	73
Authors' Addresses . . . . .	79

## 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 RESTCONF 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:

- o multiple subscriptions on a single transport session
- o support for dynamic and configured subscriptions
- o modification of an existing subscription in progress
- o per-subscription operational counters
- o negotiation of subscription parameters (through the use of hints returned as part of declined subscription requests)
- o subscription state change notifications (e.g., publisher driven suspension, parameter modification)
- o 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 a Remote Procedure Call (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:

- o 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.

- o 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. Similarly, a dynamic subscription cannot be directly modified or deleted by configuration operations. It is however possible to perform a configuration operation which indirectly impacts a dynamic subscription. By changing value of a pre-configured filter referenced by an existing dynamic subscription, the selected event records passed to a receiver might change.

Also note that transport-specific specifications based on this specification MUST detail the lifecycle of dynamic subscriptions, as well as the lifecycle of configured subscriptions (if supported).

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.
- o A subscription "stop-time" can be specified as part of a notification replay. This supports an analogous capability to the stopTime parameter of [RFC5277]. However in this specification, a "stop-time" parameter can also be applied without replay.

## 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 defined event records. An event record is an instantiation of a "notification" YANG statement. If the "notification" is defined as a child to a data node, the instantiation 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 defined 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 record 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 provides for several Quality of Service (QoS) parameters. These parameters indicate the treatment of a subscription relative to other traffic between publisher and receiver. Included are:

- o A "dscp" marking to differentiate prioritization of notification messages during network transit.
- o A "weighting" so that bandwidth proportional to this weighting can be allocated to this subscription relative to other subscriptions.
- o 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. A

publisher MUST respect the DSCP markings for subscription traffic egressing that publisher.

Different DSCP code points require different transport connections. As a result where TCP is used, a publisher which supports the "dscp" feature must ensure that a subscription's notification messages are returned within a single TCP transport session where all traffic shares the subscription's "dscp" leaf value. Where this cannot be guaranteed, any "establish subscription" RPC request SHOULD be rejected with a "dscp-unavailable" error.

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.

"Dependency" and "weight" parameters will only be respected and enforced between subscriptions that share the same "dscp" leaf value.

There are additional types over publisher capacity overload which this specification does not address within its scope. For example, the prioritization of which subscriptions have precedence when the publisher CPU is overloaded is not discussed. As a result, implementation choices will need to be made to address such considerations.

## 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 extensively 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.

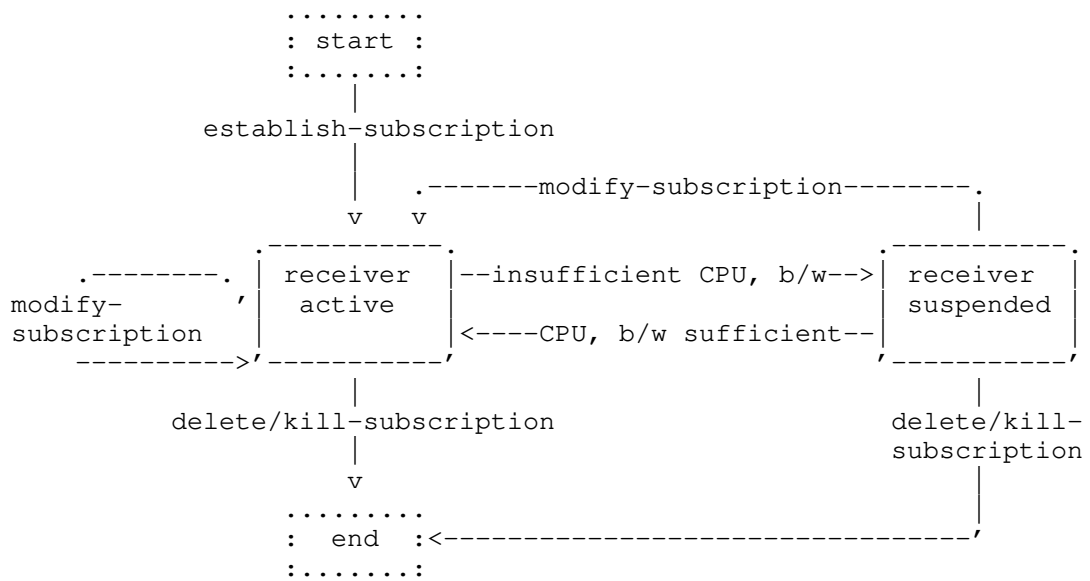


Figure 1: Publisher's state for a dynamic subscription

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.



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 event records generated in the recent past. 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.

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 variety, 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:

- o persistence across publisher reboots,
- o persistence even when transport is unavailable, and
- o 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. However if replaying missed event records is required for a configured subscription, support for dynamic subscription is highly recommended. In this case, a separate dynamic subscription can be established to retransmit the missing event records.

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:

- o 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 Virtual Routing and Forwarding (VRF) instance on which to reach receivers. This VRF is a network instance as defined within [RFC8529]. Publisher support for VRFs is optional and advertised using the "supports-vrf" feature.

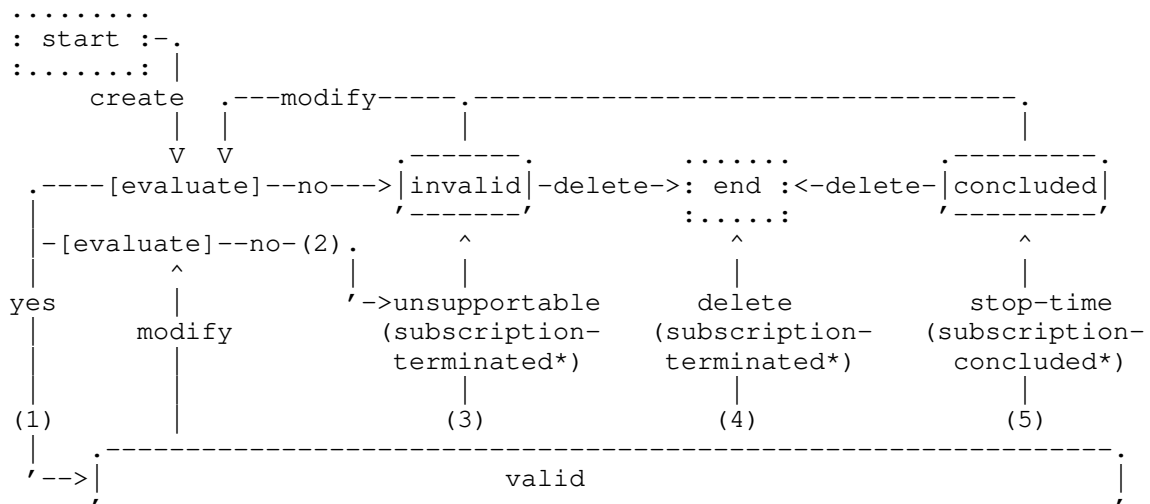
If none of the above parameters are set, notification messages MUST egress the publisher's default interface.



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.



Legend:

dotted boxes: subscription added or removed via configuration

dashed boxes: states for a subscription

```
[evaluate]: decision point on whether the subscription is supportable
```

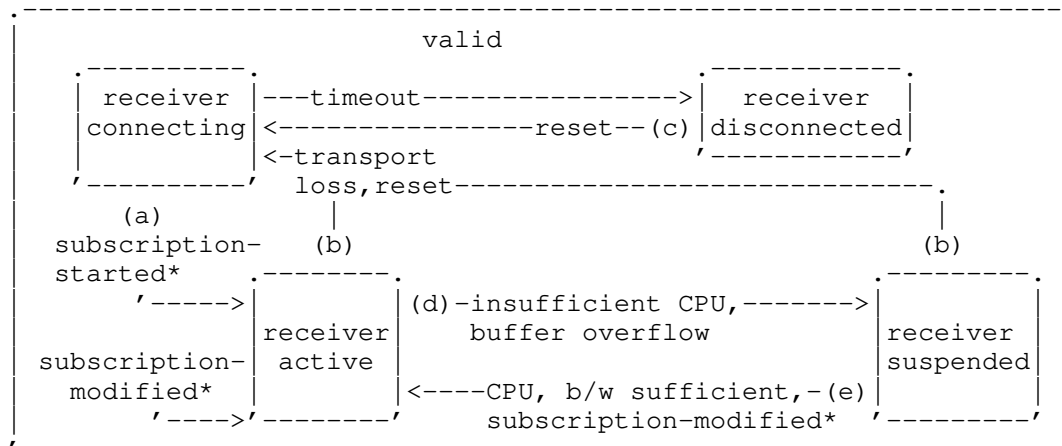
(\*) : resulting subscription state change notification

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 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.



#### 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 capacity being reached. 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 capacity being reached, 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 capacity being reached, 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 time gap where records generated on the publisher were not successfully 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

[RFC5277] Section 2.2.1 states that a notification message is to be sent to a subscriber which initiated a "create-subscription". With this specification, this [RFC5277] statement should be more broadly interpreted to mean that notification messages can also be sent to a subscriber which initiated an "establish-subscription", or a configured receiver which has been sent a "subscription-started".

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 dropped or 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.

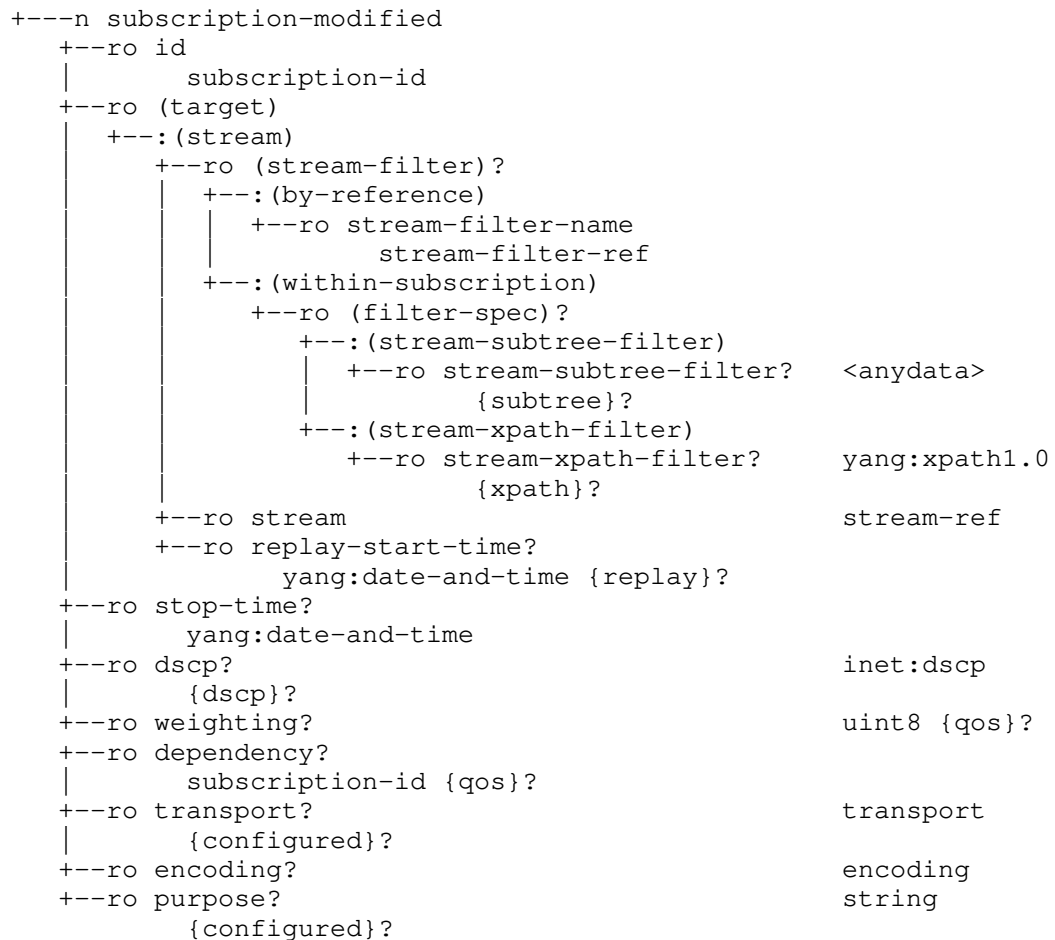


Figure 12: subscription-modified notification tree diagram

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 this is 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?
|   |   |   {configured,replay}?
|   |   |   empty
|   +--rw stop-time?
|   |   yang:date-and-time
|   +--rw dscp?
|   |   {dscp}?
|   |   inet:dscp
|   +--rw weighting?
|   |   uint8 {qos}?
|   +--rw dependency?
|   |   subscription-id {qos}?
|   +--rw transport?
|   |   {configured}?
|   |   transport
|   +--rw encoding?
|   |   encoding
|   +--rw purpose?
|   |   {configured}?
|   |   string
|   +--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?
|   |   |   {configured}?
|   |   |   enumeration
|   +--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 [RFC8529], [XPATH], [RFC6241], [RFC7049], [RFC7540], [RFC7951], [RFC7950] and [RFC8259].

[ note to the RFC Editor - please replace XXXX within this YANG model with the number of this document ]

[ 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@2019-05-06.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
      "RFC 8529: 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 2019-05-06 {
  description
    "Initial version";
  reference
    "RFC XXXX:Subscription to YANG Event Notifications";
}

/*
 * FEATURES
 */

feature configured {
  description
    "This feature indicates that configuration of subscriptions is
    supported.";
}

feature dscp {
  description
    "This feature indicates that a publisher supports the ability to
    set the DiffServ Code Point (DSCP) value in outgoing packets.";
}

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-suspended' 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. An example of a
     configurable encoding might be a new identity such as
     'encode-cbor'. Such an identity could use
     'configurable-encoding' as its base. This would allow a
     dynamic subscription encoded in JSON [RFC-8259] to request
     notification messages be encoded via CBOR [RFC-7049]. Further
     details for any specific configurable encoding would be
     explored in a transport document based on this specification.";
}

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 is the set of prefix
         and namespace pairs for all YANG modules implemented
         by the server, where the prefix is the YANG module
         name and the namespace is as defined by the
         'namespace' statement in the YANG module.

         If the leaf is encoded in XML, all namespace
         declarations in scope on the 'stream-xpath-filter'
         leaf element are added to the set of namespace
         declarations.  If a prefix found in the XML is
         already present in the set of namespace declarations,
         the namespace in the XML is used.

       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. Larger weights get
    more resources. 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. The syntax and
        semantics of this hint are implementation-specific.";
    }
}

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. The syntax and
        semantics of this hint are implementation-specific.";
    }
  }
}

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.";
  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.";
    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 is 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;
            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,
            it no longer has receivers in the 'receiver active' or
            'receiver suspended' state, 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 augmented 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 publisher 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. Note that due to the possibility of different access control permissions per receiver, each receiver may actually get a different set of event records.

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. Beyond this, 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.

A specific transport specification MUST identify any encoding supported. Where a configured subscription's transport allows different encodings, the specification MUST identify the default encoding.

A subscriber which includes a "dscp" leaf within an "establish-subscription" request will need to understand and consider what the corresponding DSCP value represents within the domain of the publisher.

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.

With configured subscriptions, one or more publishers could be used to overwhelm a receiver. To counter this, 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, it may indicate that an attacker has done something that has momentarily disrupted receiver connectivity. To acquire events lost during this interval, the receiver SHOULD retrieve any event records generated since the last event record was received. This can be accomplished 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.

The replay mechanisms described in Section 2.4.2.1 and Section 2.5.6 provides access to historical event records. By design, the access control model that protects these records could enable subscribers to view data to which they were not authorized at the time of collection.

Using DNS names for configured subscription receiver "name" lookup can cause situations where the name resolves unexpectedly differently on the publisher, so the recipient would be different than expected.

An attacker that can cause the publisher to use an incorrect time can induce message replay by setting the time in the past, and introduce a risk of message loss by setting the time in the future.

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. Improper configuration

could provide a receiver with information leakage consisting of the dropping of event records.

- 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

- [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>>.
- [RFC8529] Berger, L., Hopps, C., Lindem, A., Bogdanovic, D., and X. Liu, "YANG Data Model for Network Instances", RFC 8529, DOI 10.17487/RFC8529, March 2019, <<https://www.rfc-editor.org/info/rfc8529>>.
- [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 2019, <<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 2019, <<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 2019, <<https://datatracker.ietf.org/doc/draft-ietf-netconf-yang-push/>>.
- [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>>.
- [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>>.
- [RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <<https://www.rfc-editor.org/info/rfc8259>>.
- [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. This example is not intended to be a complete transport model. 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)

### v25 - v26

- o Tweaks from Alissa Cooper's review, and Benjamin Kaduk's discuss.
- o Magnus' review help refine the words on several 'overload' considerations. And a couple of QoS requirements were clarified.
- o Note on interpreting RFC-5277 so that notification messages can follow establish-subscription RPCs.
- o draft-ietf-rtgwg-ni-model updated to RFC-8529

### v24 - v25

- o Replay security consideration added based on Roman Danyliw's discuss
- o Spelling fixes, acronyms expanded
- o Tweaks and updates based Benjamin Kaduk's comments. This includes the adding of clarifying security considerations, a couple of clarifications in the YANG definitions, and ensuring a fuller set of transport specification requirements are defined in 5.3.

### v23 - v24

- o Per Benjamin Kaduk's discuss, adjusted IPR to pre5378Trust200902
- o Tweaks from Chris Lonvick's IESG review. This includes moving a paragraph from Security Considerations into a sentence within Implementation Considerations.
- o Tweaks from Wesley Eddy DSCP description

v22 - v23

- o During the YANG Doctor review, feature dscp support was refined to avoid the out-of-order delivery of packets in a single TCP session.

v21 - v22

- o YANG Dr definition clarifications. This includes refined text on: (a) stop-time can be used without replay, (b) a separate dynamic subscription for replay, (c) subscription state change notifications can't be dropped, more details on "enum concluded" and (d) more text on configurable-encoding leaf (which adds two informative references). There also was one minor tweak in the YANG model. The stream description leaf had "mandatory true" removed.

v20 - v21

- o Editorial change in Section 1.3 requested by Qin's Shepherd review of NETCONF-Notif and RESTCONF-Notif. Basically extra text was added further describing that dynamic subscriptions can have state change notifications.

v18 - v20

- o XPath-stream-filter YANG object definition updated based on NETMOD discussions.

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 where 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](mailto:evoit@cisco.com)

Alexander Clemm  
Huawei

Email: [ludwig@clemm.org](mailto:ludwig@clemm.org)

Alberto Gonzalez Prieto  
Microsoft

Email: [alberto.gonzalez@microsoft.com](mailto:alberto.gonzalez@microsoft.com)

Einar Nilsen-Nygaard  
Cisco Systems

Email: [einarnn@cisco.com](mailto:einarnn@cisco.com)

Ambika Prasad Tripathy  
Cisco Systems

Email: [ambtripa@cisco.com](mailto:ambtripa@cisco.com)

NETCONF Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: September 14, 2017

K. Watsen  
Juniper Networks  
G. Wu  
Cisco Systems  
March 13, 2017

TLS Client and Server Models  
draft-ietf-netconf-tls-client-server-02

Abstract

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

Editorial Note (To be removed by RFC Editor)

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

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

- o I-D.ietf-netconf-keystore

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

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

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

- o "2017-03-13" --> the publication date of this draft

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

- o Appendix A. Change Log

- o Appendix B. Open Issues

## 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 <http://datatracker.ietf.org/drafts/current/>.

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

This Internet-Draft will expire on September 14, 2017.

## Copyright Notice

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

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

## Table of Contents

1. Introduction . . . . .	3
1.1. Terminology . . . . .	3
1.2. Tree Diagrams . . . . .	4
2. The TLS Client Model . . . . .	4
2.1. Tree Diagram . . . . .	4
2.2. Example Usage . . . . .	5
2.3. YANG Model . . . . .	5
3. The TLS Server Model . . . . .	8
3.1. Tree Diagram . . . . .	8
3.2. Example Usage . . . . .	9
3.3. YANG Model . . . . .	9
4. The TLS Common Model . . . . .	12



4.1. Tree Diagram . . . . .	13
4.2. Example Usage . . . . .	13
4.3. YANG Model . . . . .	13
5. Security Considerations . . . . .	21
6. IANA Considerations . . . . .	22
6.1. The IETF XML Registry . . . . .	22
6.2. The YANG Module Names Registry . . . . .	22
7. Acknowledgements . . . . .	22
8. References . . . . .	23
8.1. Normative References . . . . .	23
8.2. Informative References . . . . .	24
Appendix A. Change Log . . . . .	25
A.1. server-model-09 to 00 . . . . .	25
A.2. 00 to 01 . . . . .	25
A.3. 01 to 02 . . . . .	25
Appendix B. Open Issues . . . . .	25
Authors' Addresses . . . . .	25

## 1. Introduction

This document defines three YANG [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 enables 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 grouping for the TLS parts it provides, while adding data nodes for the TCP-level call-home configuration.

### 1.1. Terminology

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

## 1.2. Tree Diagrams

A simplified graphical representation of the data models is used in this document. The meaning of the symbols in these diagrams is as follows:

- o Brackets "[" and "]" enclose list keys.
- o Braces "{" and "}" enclose feature names, and indicate that the named feature must be present for the subtree to be present.
- o Abbreviations before data node names: "rw" means configuration (read-write) and "ro" state data (read-only).
- o Symbols after data node names: "?" means an optional node, "!" means a presence container, and "\*" denotes a list and leaf-list.
- o Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- o Ellipsis ("...") stands for contents of subtrees that are not shown.

## 2. The TLS Client Model

The TLS client model presented in this section contains one YANG grouping, to just configure the TLS client omitting, for instance, any configuration for which IP address or port the client should connect to.

This grouping references data nodes defined by the keystore model [I-D.ietf-netconf-keystore]. For instance, a reference to the keystore model is made to indicate which trusted CA certificate a client should use to authenticate the server's certificate.

### 2.1. Tree Diagram

The following tree diagram presents the data model for the grouping defined in the ietf-tls-client module. Please see Section 1.2 for tree diagram notation.

```

module: ietf-tls-client
groupings:
  tls-client-grouping
    +----- server-auth
    |   +----- trusted-ca-certs?
    |   |           -> /ks:keystore/trusted-certificates/name
    |   +----- trusted-server-certs?
    |   |           -> /ks:keystore/trusted-certificates/name
    +----- client-auth
    |   +----- (auth-type)?
    |   |   +---:(certificate)
    |   |   +----- certificate?   leafref
    +----- hello-params {tls-client-hello-params-config}?
    |   +----- tls-versions
    |   |   +----- tls-version*   identityref
    +----- cipher-suites
    |   +----- cipher-suite*   identityref

```

## 2.2. Example Usage

This section shows how it would appear if the `tls-client-grouping` were populated with some data. This example is consistent with the examples presented in Section 2.2 of [I-D.ietf-netconf-keystore].

```

<!-- hypothetical example, as groupings don't have instance data -->
<tls-client xmlns="urn:ietf:params:xml:ns:yang:ietf-tls-client">

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

  <!-- how this client will authenticate itself to the server -->
  <client-auth>
    <certificate>builtin-idevid-cert</certificate>
  </client-auth>

</tls-client>

```

## 2.3. YANG Model

This YANG module has a normative references to [RFC6991] and [I-D.ietf-netconf-keystore].

```
<CODE BEGINS> file "ietf-tls-client@2017-03-13.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 tlscom;
    revision-date 2017-03-13; // stable grouping definitions
    reference
      "RFC XXXX: TLS Client and Server Models";
  }

  import ietf-keystore {
    prefix ks;
    reference
      "RFC YYYY: Keystore Model";
  }

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

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

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

  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) 2014 IETF Trust and the persons identified as
    authors of the code. All rights reserved.

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

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

  revision "2017-03-13" {
```

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

  feature tls-client-hello-params-config {
    description
      "TLS hello message parameters are configurable on a TLS
      client.";
  }

  grouping tls-client-grouping {
    description
      "A reusable grouping for configuring a TLS client without
      any consideration for how an underlying TCP session is
      established.";

    container server-auth {
      description
        "Trusted server identities.";

      leaf trusted-ca-certs {
        type leafref {
          path "/ks:keystore/ks:trusted-certificates/ks:name";
        }
        description
          "A reference to a list of certificate authority (CA)
          certificates used by the TLS client to authenticate
          TLS server certificates.";
      }

      leaf trusted-server-certs {
        type leafref {
          path "/ks:keystore/ks:trusted-certificates/ks:name";
        }
        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 trusted server certificate.";
      }
    }
  }

  container client-auth {
    description
      "The credentials used by the client to authenticate to
      the TLS server.";
```

```
    choice auth-type {
      description
        "The authentication type.";
      leaf certificate {
        type leafref {
          path "/ks:keystore/ks:keys/ks:key/ks:certificates"
            + "/ks:certificate/ks:name";
        }
        description
          "A certificates to be used for user authentication.";
      }
    }
  }
}

container hello-params {
  if-feature tls-client-hello-params-config;
  uses tlscom:hello-params-grouping;
  description
    "Configurable parameters for the TLS hello message.";
}

} // end tls-client-grouping
}
```

<CODE ENDS>

### 3. The TLS Server Model

The TLS server model presented in this section contains one YANG grouping, for just the TLS-level configuration omitting, for instance, configuration for which ports to open to listen for connections on.

This grouping references data nodes defined by the keystore model [I-D.ietf-netconf-keystore]. For instance, a reference to the keystore model is made to indicate which certificate a server should present.

#### 3.1. Tree Diagram

The following tree diagram presents the data model for the grouping defined in the ietf-tls-server module. Please see Section 1.2 for tree diagram notation.

```

module: ietf-tls-server
groupings:
  tls-server-grouping
    +---- certificates
    |   +---- certificate* [name]
    |   |   +---- name?    leafref
    +---- client-auth
    |   +---- trusted-ca-certs?
    |   |   -> /ks:keystore/trusted-certificates/name
    |   +---- trusted-client-certs?
    |   |   -> /ks:keystore/trusted-certificates/name
    +---- hello-params {tls-server-hello-params-config}?
    |   +---- tls-versions
    |   |   +---- tls-version*    identityref
    +---- cipher-suites
    |   +---- cipher-suite*    identityref

```

### 3.2. Example Usage

This section shows how it would appear if the `tls-server-grouping` were populated with some data. This example is consistent with the examples presented in Section 2.2 of [I-D.ietf-netconf-keystore].

```
<!-- hypothetical example, groupings don't have instance data -->
```

```

<tls-server
  xmlns="urn:ietf:params:xml:ns:yang:ietf-tls-server">
  <certificates>
    <certificate>
      <name>tls-ec-cert</name>
    </certificate>
  </certificates>
  <client-auth>
    <trusted-ca-certs>deployment-specific-ca-certs</trusted-ca-certs>
    <trusted-client-certs>explicitly-trusted-client-certs</trusted-client-certs>
  </client-auth>
</tls-server>

```

### 3.3. YANG Model

This YANG module has a normative references to [RFC6991], and [I-D.ietf-netconf-keystore].

```

<CODE BEGINS> file "ietf-tls-server@2017-03-13.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 tlscom;
  revision-date 2017-03-13; // stable grouping definitions
  reference
    "RFC XXXX: TLS Client and Server Models";
}

import ietf-keystore {
  prefix ks;
  reference
    "RFC YYYY: Keystore Model";
}

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

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

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

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) 2014 IETF Trust and the persons identified as
  authors of the code. All rights reserved.

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

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

revision "2017-03-13" {
  description
```



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

feature tls-server-hello-params-config {
  description
    "TLS hello message parameters are configurable on a TLS
    server.";
}

// grouping
grouping tls-server-grouping {
  description
    "A reusable grouping for configuring a TLS server without
    any consideration for how underlying TCP sessions are
    established.";
  container certificates {
    description
      "The list of 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";
    list certificate {
      key name;
      min-elements 1;
      description
        "An unordered list of certificates the TLS server can pick
        from when sending its Server Certificate message.";
      reference
        "RFC 5246: The TLS Protocol, Section 7.4.2";
      leaf name {
        type leafref {
          path "/ks:keystore/ks:keys/ks:key/ks:certificates/"
            + "ks:certificate/ks:name";
        }
        description
          "The name of the certificate in the keystore.";
      }
    }
  }
}

container client-auth {
  description
    "A reference to a list of trusted certificate authority (CA)
    certificates and a reference to a list of trusted client
```

```
        certificates.";
    leaf trusted-ca-certs {
        type leafref {
            path "/ks:keystore/ks:trusted-certificates/ks:name";
        }
        description
            "A reference to a list of certificate authority (CA)
            certificates used by the TLS server to authenticate
            TLS client certificates.";
    }

    leaf trusted-client-certs {
        type leafref {
            path "/ks:keystore/ks:trusted-certificates/ks:name";
        }
        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 trusted client certificate.";
    }
}

container hello-params {
    if-feature tls-server-hello-params-config;
    uses tlscom:hello-params-grouping;
    description
        "Configurable parameters for the TLS hello message.";
}

} // end tls-server-grouping
}
```

<CODE ENDS>

#### 4. 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 security policies.

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.

#### 4.1. Tree Diagram

The following tree diagram presents the data model for the grouping defined in the ietf-tls-common module. Please see Section 1.2 for tree diagram notation.

```
module: ietf-tls-common
  groupings:
    hello-params-grouping
      +----- tls-versions
      |   +----- tls-version*   identityref
      +----- cipher-suites
      |   +----- cipher-suite*   identityref
```

#### 4.2. Example Usage

This section shows how it would appear if the transport-params-grouping were populated with some data.

```
<!-- hypothetical example, as groupings don't have instance data -->
<hello-params xmlns="urn:ietf:params:xml:ns:yang:ietf-tls-common">

  <tls-versions>
    <tls-version>tls-1.2</tls-version>
  </tls-versions>
  <cipher-suites>
    <cipher-suite>ecdhe-rsa-with-3des-edc-cbc-sha</cipher-suite>
    <cipher-suite>dhe-rsa-with-aes-128-cbc-sha</cipher-suite>
    <cipher-suite>rsa-with-aes-128-cbc-sha</cipher-suite>
    <cipher-suite>rsa-with-3des-edc-cbc-sha</cipher-suite>
  </cipher-suites>

</hello-params>
```

#### 4.3. YANG Model

This YANG module has a normative references to [RFC4492], [RFC5246], [RFC5288], and [RFC5289].

```
<CODE BEGINS> file "ietf-tls-common@2017-03-13.yang"

module ietf-tls-common {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-tls-common";
  prefix "tlscom";

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

  contact
    "WG Web:    <http://tools.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) 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 "2017-03-13" {
    description
      "Initial version";
    reference
      "RFC XXXX: TLS Client and Server Models";
  }

  // features
  feature tls-ecc {
```

```
    description
        "Elliptic Curve Cryptography (ECC) is supported for TLS.";
    reference
        "RFC 4492: 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-1.2 {
    base tls-version-base;
    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 4492: 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 4492: 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 4492: 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.  For
      configurable parameters, a zero-element leaf-list indicates the
      system default configuration for that parameter.";
  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
      "Allowed TLS protocol versions.";
  }
}

container cipher-suites {
  description
    "Parameters regarding cipher suites.";
  leaf-list cipher-suite {
    type identityref {
      base cipher-suite-base;
    }
  }
  ordered-by user;
  description
    "Cipher suites in order of descending preference.";
}
}
```

```
}  
}
```

```
<CODE ENDS>
```

## 5. 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) [RFC6536] 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  
   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

## 6. IANA Considerations

### 6.1. The IETF XML Registry

This document registers three URIs in 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.

### 6.2. The YANG Module Names Registry

This document registers three YANG modules in the YANG Module Names registry [RFC7950]. Following the format in [RFC7950], 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: tlss  
reference: RFC XXXX

## 7. Acknowledgements

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

## 8. References

### 8.1. Normative References

- [I-D.ietf-netconf-keystore]  
Watsen, K. and G. Wu, "Keystore Model", draft-ietf-netconf-keystore-00 (work in progress), October 2016.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC4492] Blake-Wilson, S., Bolyard, N., Gupta, V., Hawk, C., and B. Moeller, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)", RFC 4492, DOI 10.17487/RFC4492, May 2006, <<http://www.rfc-editor.org/info/rfc4492>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<http://www.rfc-editor.org/info/rfc5246>>.
- [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, <<http://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, <<http://www.rfc-editor.org/info/rfc5289>>.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <<http://www.rfc-editor.org/info/rfc6536>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<http://www.rfc-editor.org/info/rfc6991>>.
- [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, <<http://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, <<http://www.rfc-editor.org/info/rfc7950>>.

## 8.2. Informative References

- [RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <<http://www.rfc-editor.org/info/rfc2818>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<http://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, <<http://www.rfc-editor.org/info/rfc6241>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<http://www.rfc-editor.org/info/rfc8040>>.
- [RFC8071] Watsen, K., "NETCONF Call Home and RESTCONF Call Home", RFC 8071, DOI 10.17487/RFC8071, February 2017, <<http://www.rfc-editor.org/info/rfc8071>>.

## Appendix A. Change Log

## A.1. server-model-09 to 00

- o This draft was split out from draft-ietf-netconf-server-model-09.
- o Noted that '0.0.0.0' and ':::' might have special meanings.

## A.2. 00 to 01

- o Renamed "keychain" to "keystore".

## A.3. 01 to 02

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

## Appendix B. Open Issues

Please see: <https://github.com/netconf-wg/tls-client-server/issues>.

## Authors' Addresses

Kent Watsen  
Juniper Networks

EMail: [kwatsen@juniper.net](mailto:kwatsen@juniper.net)

Gary Wu  
Cisco Systems

EMail: [garywu@cisco.com](mailto:garywu@cisco.com)

NETCONF Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: 8 September 2022

K. Watsen  
Watsen Networks  
7 March 2022

YANG Groupings for TLS Clients and TLS Servers  
draft-ietf-netconf-tls-client-server-27

Abstract

This document defines three YANG 1.1 modules: the first defines features and groupings common to both TLS clients and TLS servers, the second defines a grouping for a generic TLS client, and the third defines a grouping for a generic TLS server.

Editorial Note (To be removed by RFC Editor)

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

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

- \* AAAA --> the assigned RFC value for draft-ietf-netconf-crypto-types
- \* BBBB --> the assigned RFC value for draft-ietf-netconf-trust-anchors
- \* CCCC --> the assigned RFC value for draft-ietf-netconf-keystore
- \* DDDD --> the assigned RFC value for draft-ietf-netconf-tcp-client-server
- \* FFFF --> 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:

- \* 2022-03-07 --> the publication date of this draft

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

- \* Appendix B. Change Log



## Status of This Memo

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

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

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

This Internet-Draft will expire on 8 September 2022.

## Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

1. Introduction . . . . .	4
1.1. Relation to other RFCs . . . . .	4
1.2. Specification Language . . . . .	6
1.3. Adherence to the NMDA . . . . .	6
1.4. Conventions . . . . .	6
2. The "ietf-tls-common" Module . . . . .	7
2.1. Data Model Overview . . . . .	7
2.2. Example Usage . . . . .	9
2.3. YANG Module . . . . .	9
3. The "ietf-tls-client" Module . . . . .	14
3.1. Data Model Overview . . . . .	14
3.2. Example Usage . . . . .	17
3.3. YANG Module . . . . .	21
4. The "ietf-tls-server" Module . . . . .	33
4.1. Data Model Overview . . . . .	33

4.2. Example Usage . . . . .	35
4.3. YANG Module . . . . .	39
5. Security Considerations . . . . .	51
5.1. The "iana-tls-cipher-suite-algs" Module . . . . .	51
5.2. The "ietf-tls-common" YANG Module . . . . .	51
5.3. The "ietf-tls-client" YANG Module . . . . .	52
5.4. The "ietf-tls-server" YANG Module . . . . .	53
6. IANA Considerations . . . . .	53
6.1. The "IETF XML" Registry . . . . .	53
6.2. The "YANG Module Names" Registry . . . . .	54
6.3. The "iana-tls-cipher-suite-algs" Module . . . . .	54
7. References . . . . .	55
7.1. Normative References . . . . .	55
7.2. Informative References . . . . .	56
Appendix A. YANG Modules for IANA . . . . .	59
A.1. Initial Module for the "TLS Cipher Suites" Registry . . . . .	59
A.1.1. Data Model Overview . . . . .	59
A.1.2. Example Usage . . . . .	60
A.1.3. YANG Module . . . . .	61
Appendix B. Change Log . . . . .	139
B.1. 00 to 01 . . . . .	139
B.2. 01 to 02 . . . . .	139
B.3. 02 to 03 . . . . .	139
B.4. 03 to 04 . . . . .	140
B.5. 04 to 05 . . . . .	140
B.6. 05 to 06 . . . . .	140
B.7. 06 to 07 . . . . .	140
B.8. 07 to 08 . . . . .	140
B.9. 08 to 09 . . . . .	141
B.10. 09 to 10 . . . . .	141
B.11. 10 to 11 . . . . .	141
B.12. 11 to 12 . . . . .	141
B.13. 12 to 13 . . . . .	142
B.14. 12 to 13 . . . . .	142
B.15. 13 to 14 . . . . .	142
B.16. 14 to 15 . . . . .	142
B.17. 15 to 16 . . . . .	142
B.18. 16 to 17 . . . . .	143
B.19. 17 to 18 . . . . .	143
B.20. 18 to 19 . . . . .	143
B.21. 19 to 20 . . . . .	144
B.22. 20 to 21 . . . . .	144
B.23. 21 to 22 . . . . .	144
B.24. 22 to 23 . . . . .	145
B.25. 23 to 24 . . . . .	145
B.26. 24 to 25 . . . . .	145
B.27. 25 to 26 . . . . .	145
B.28. 26 to 27 . . . . .	145

Acknowledgements	146
Contributors	146
Author's Address	146

## 1. Introduction

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

Any version of TLS may be configured. TLS 1.0 [RFC2246] and TLS 1.1 [RFC4346] are historic and hence the YANG "feature" statements enabling them are marked "status obsolete". TLS 1.2 [RFC5246] is obsoleted by TLS 1.3 [RFC8446] but still in common use, and hence its "feature" statement is marked "status deprecated". All the feature statements for 1.0, 1.1, and 1.3 have "description" statements stating that it is NOT RECOMMENDED to enable obsolete protocol versions.

It is intended that the YANG groupings will be used by applications needing to configure TLS client and server protocol stacks. For instance, these groupings are used to help define the data model for HTTPS [RFC2818] and NETCONF over TLS [RFC7589] based clients and servers in [I-D.ietf-netconf-http-client-server] and [I-D.ietf-netconf-netconf-client-server] respectively.

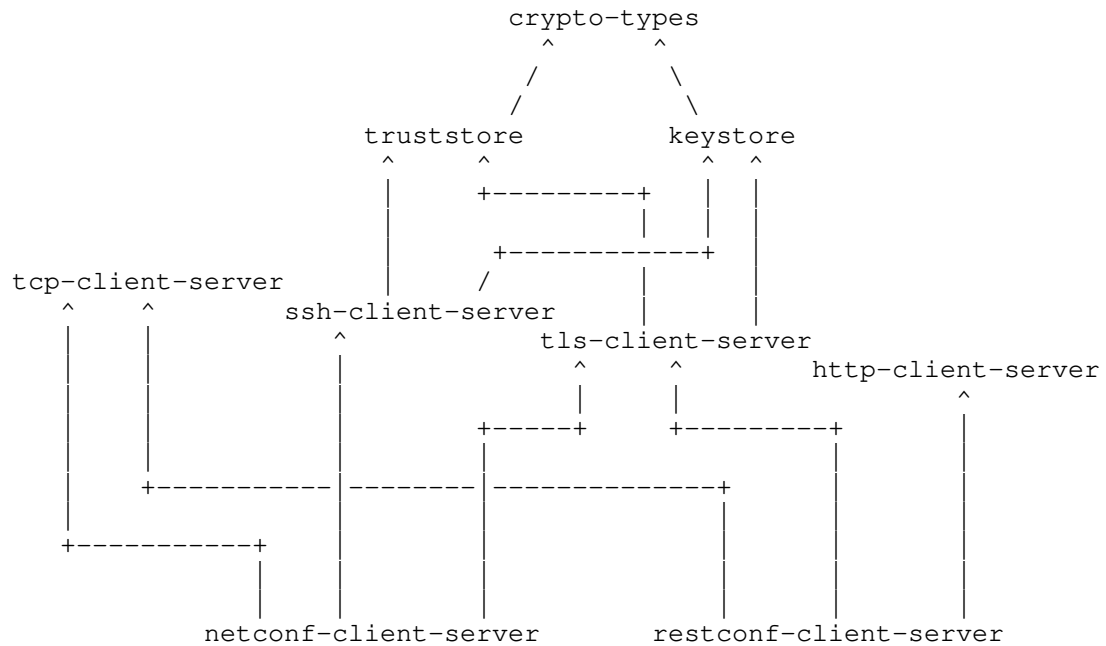
The client and server YANG modules in this document each define one grouping, which is focused on just TLS-specific configuration, and specifically avoids any transport-level configuration, such as what ports to listen-on or connect-to. This affords applications the opportunity to define their own strategy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

### 1.1. Relation to other RFCs

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

The modules have been defined in a modular fashion to enable their use by other efforts, some of which are known to be in progress at the time of this writing, with many more expected to be defined in time.

The normative dependency relationship between the various RFCs in the collection is presented in the below diagram. The labels in the diagram represent the primary purpose provided by each RFC. Hyperlinks to each RFC are provided below the diagram.



Label in Diagram	Originating RFC
crypto-types	[I-D.ietf-netconf-crypto-types]
truststore	[I-D.ietf-netconf-trust-anchors]
keystore	[I-D.ietf-netconf-keystore]
tcp-client-server	[I-D.ietf-netconf-tcp-client-server]
ssh-client-server	[I-D.ietf-netconf-ssh-client-server]
tls-client-server	[I-D.ietf-netconf-tls-client-server]
http-client-server	[I-D.ietf-netconf-http-client-server]
netconf-client-server	[I-D.ietf-netconf-netconf-client-server]
restconf-client-server	[I-D.ietf-netconf-restconf-client-server]

Table 1: Label to RFC Mapping

## 1.2. Specification Language

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

## 1.3. Adherence to the NMDA

This document is compliant with the Network Management Datastore Architecture (NMDA) [RFC8342]. For instance, as described in [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore], trust anchors and keys installed during manufacturing are expected to appear in <operational>.

## 1.4. Conventions

Various examples used in this document use a placeholder value for binary data that has been base64 encoded (e.g., "BASE64VALUE="). This placeholder value is used as real base64 encoded structures are often many lines long and hence distracting to the example being presented.

## 2. The "ietf-tls-common" Module

The TLS common model presented in this section contains features and groupings common to both TLS clients and TLS servers. The "hello-params-grouping" grouping can be used to configure the list of TLS algorithms permitted by the TLS client or TLS server. The lists of algorithms are ordered such that, if multiple algorithms are permitted by the client, the algorithm that appears first in its list that is also permitted by the server is used for the TLS transport layer connection. The ability to restrict the algorithms allowed is provided in this grouping for TLS clients and TLS servers that are capable of doing so and may serve to make TLS clients and TLS servers compliant with local security policies. This model supports both TLS1.2 [RFC5246] and TLS 1.3 [RFC8446].

Thus, in order to support both TLS1.2 and TLS1.3, the cipher-suites part of the "hello-params-grouping" 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.

### 2.1. Data Model Overview

This section provides an overview of the "ietf-tls-common" module in terms of its features, identities and groupings.

#### 2.1.1. Features

The following diagram lists all the "feature" statements defined in the "ietf-tls-common" module:

Features:

```
+-- tls-1_0
+-- tls-1_1
+-- tls-1_2
+-- tls-1_3
+-- hello-params
```

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

#### 2.1.2. Identities

The following diagram illustrates the relationship amongst the "identity" statements defined in the "ietf-tls-common" module:

## Identities:

```
+-- tls-version-base
  +-- tls-1.0
  +-- tls-1.1
  +-- tls-1.2
  +-- tls-1.3
```

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

## Comments:

- \* The diagram shows that there are two base identities.
- \* One base identity is used to specific TLS versions, while the other is used to specify cipher-suites.
- \* These base identities are "abstract", in the object oriented programming sense, in that they only define a "class" of things, rather than a specific thing.

## 2.1.3. Groupings

The "ietf-tls-common" module defines the following "grouping" statement:

```
* hello-params-grouping
```

This grouping is presented in the following subsection.

## 2.1.3.1. The "hello-params-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "hello-params-grouping" grouping:

```
grouping hello-params-grouping
  +-- tls-versions
  |   +-- tls-version*    identityref
  +-- cipher-suites
      +-- cipher-suite*   identityref
```

## Comments:

- \* This grouping is used by both the "tls-client-grouping" and the "tls-server-grouping" groupings defined in Section 3.1.2.1 and Section 4.1.2.1, respectively.
- \* This grouping enables client and server configurations to specify the TLS versions and cipher suites that are to be used when establishing TLS sessions.

- \* The "cipher-suites" list is "ordered-by user".

#### 2.1.4. Protocol-accessible Nodes

The "ietf-tls-common" module defines only "grouping" statements that are used by other modules to instantiate protocol-accessible nodes.

#### 2.2. Example Usage

This section shows how it would appear if the "hello-params-grouping" grouping were populated with some data.

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

```
<!-- The outermost element below doesn't exist in the data model. -->
<!-- It simulates if the "grouping" were a "container" instead. -->
```

```
<hello-params
  xmlns="urn:ietf:params:xml:ns:yang:ietf-tls-common"
  xmlns:tlscmn="urn:ietf:params:xml:ns:yang:ietf-tls-common"
  xmlns:tlscsa="urn:ietf:params:xml:ns:yang:iana-tls-cipher-suite-a\
lgs">
  <tls-versions>
    <tls-version>tlscmn:tls-1.1</tls-version>
    <tls-version>tlscmn:tls-1.2</tls-version>
  </tls-versions>
  <cipher-suites>
    <cipher-suite>tlscsa:tls-ecdh-e-ecdsa-with-aes-256-cbc-sha</ciphe\
r-suite>
    <cipher-suite>tlscsa:tls-dhe-rsa-with-aes-128-cbc-sha256</cipher\
-suite>
    <cipher-suite>tlscsa:tls-rsa-with-3des-edc-cbc-sha</cipher-suite>
  </cipher-suites>
</hello-params>
```

#### 2.3. YANG Module

This YANG module has a normative references to [RFC4346], [RFC5288], [RFC5289], [RFC8422], and FIPS PUB 180-4.

This YANG module has a informative references to [RFC2246], [RFC4346], [RFC5246], and [RFC8446].

```
<CODE BEGINS> file "ietf-tls-common@2022-03-07.yang"
```



```
module ietf-tls-common {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-tls-common";
  prefix tlscmn;

  import iana-tls-cipher-suite-algs {
    prefix tlscsa;
    reference
      "RFC FFFF: YANG Groupings for TLS Clients and SSH Servers";
  }

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

  contact
    "WG List:  NETCONF WG list <mailto:netconf@ietf.org>
    WG Web:    https://datatracker.ietf.org/wg/netconf
    Author:    Kent Watsen <mailto:kent+ietf@watsen.net>
    Author:    Gary Wu <mailto:garywu@cisco.com>
    Author:    Jeff Hartley <mailto:jeff.hartley@commscope.com>";

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

    Copyright (c) 2021 IETF Trust and the persons identified
    as authors of the code. All rights reserved.

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

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

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

  revision 2022-03-07 {
    description
```

```
    "Initial version";
  reference
    "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
}

// Features

feature tls10 {
  status "obsolete";
  description
    "TLS Protocol Version 1.0 is supported.  TLS 1.0 is obsolete
    and thus it is NOT RECOMMENDED to enable this feature.";
  reference
    "RFC 2246: The TLS Protocol Version 1.0";
}

feature tls11 {
  status "obsolete";
  description
    "TLS Protocol Version 1.1 is supported.  TLS 1.1 is obsolete
    and thus it is NOT RECOMMENDED to enable this feature.";
  reference
    "RFC 4346: The Transport Layer Security (TLS) Protocol
    Version 1.1";
}

feature tls12 {
  status "deprecated";
  description
    "TLS Protocol Version 1.2 is supported  TLS 1.2 is obsolete
    and thus it is NOT RECOMMENDED to enable this feature.";
  reference
    "RFC 5246: The Transport Layer Security (TLS) Protocol
    Version 1.2";
}

feature tls13 {
  description
    "TLS Protocol Version 1.3 is supported.";
  reference
    "RFC 8446: The Transport Layer Security (TLS)
    Protocol Version 1.3";
}

feature hello-params {
  description
    "TLS hello message parameters are configurable.";
}
```

```
// Identities

identity tls-version-base {
  description
    "Base identity used to identify TLS protocol versions.";
}

identity tls10 {
  if-feature "tls10";
  base tls-version-base;
  status "obsolete";
  description
    "TLS Protocol Version 1.0.";
  reference
    "RFC 2246: The TLS Protocol Version 1.0";
}

identity tls11 {
  if-feature "tls11";
  base tls-version-base;
  status "obsolete";
  description
    "TLS Protocol Version 1.1.";
  reference
    "RFC 4346: The Transport Layer Security (TLS) Protocol
      Version 1.1";
}

identity tls12 {
  if-feature "tls12";
  base tls-version-base;
  status "deprecated";
  description
    "TLS Protocol Version 1.2.";
  reference
    "RFC 5246: The Transport Layer Security (TLS) Protocol
      Version 1.2";
}

identity tls13 {
  if-feature "tls13";
  base tls-version-base;
  description
    "TLS Protocol Version 1.3.";
  reference
    "RFC 8446: The Transport Layer Security (TLS)
      Protocol Version 1.3";
}
```

```
typedef epsk-supported-hash {
  type enumeration {
    enum sha-256 {
      description
        "The SHA-256 Hash.";
    }
    enum sha-384 {
      description
        "The SHA-384 Hash.";
    }
  }
  description
    "As per Section 4.2.11 of RFC 8446, the hash algorithm
    supported by an instance of an External Pre-Shared
    Key (EPSK).";
  reference
    "RFC 8446: The Transport Layer Security (TLS)
    Protocol Version 1.3
    I-D.ietf-tls-external-psk-importer: Importing
    External PSKs for TLS
    I-D.ietf-tls-external-psk-guidance: Guidance
    for External PSK Usage in 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
    RFC 8446: The Transport Layer Security (TLS) Protocol
    Version 1.3";
  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 tlscsa:cipher-suite-alg-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 FFFF for  
        valid combinations.  
  
        If this leaf-list is not configured (has zero elements)  
        the acceptable cipher suites are implementation-  
        defined.";  
      reference  
        "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";  
    }  
  }  
} // hello-params-grouping  
  
}  
  
<CODE ENDS>
```

### 3. The "ietf-tls-client" Module

This section defines a YANG 1.1 [RFC7950] module called "ietf-tls-client". A high-level overview of the module is provided in Section 3.1. Examples illustrating the module's use are provided in Examples (Section 3.2). The YANG module itself is defined in Section 3.3.

#### 3.1. Data Model Overview

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

##### 3.1.1. Features

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

## Features:

```
+-- tls-client-keepalives
+-- client-ident-x509-cert
+-- client-ident-raw-public-key
+-- client-ident-psk
+-- server-auth-x509-cert
+-- server-auth-raw-public-key
+-- server-auth-psk
```

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

## 3.1.2. Groupings

The "ietf-tls-client" module defines the following "grouping" statement:

```
* tls-client-grouping
```

This grouping is presented in the following subsection.

## 3.1.2.1. The "tls-client-grouping" Grouping

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

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

```

grouping tls-client-grouping
+-- client-identity!
|   +-- (auth-type)
|       +--:(certificate) {client-ident-x509-cert}?
|           +-- certificate
|               +---u ks:local-or-keystore-end-entity-cert-with-key-\
grouping
|       +--:(raw-public-key) {client-ident-raw-public-key}?
|           +-- raw-private-key
|               +---u ks:local-or-keystore-asymmetric-key-grouping
+--:(tls12-psk) {client-ident-tls12-psk}?
|   +-- tls12-psk
|       +---u ks:local-or-keystore-symmetric-key-grouping
|       +-- id?
|           string
+--:(tls13-epsk) {client-ident-tls13-epsk}?
|   +-- tls13-epsk
|       +---u ks:local-or-keystore-symmetric-key-grouping
|       +-- external-identity
|           |   string
|       +-- hash
|           |   tlscmn:epsk-supported-hash
|       +-- context?
|           |   string
|       +-- target-protocol?
|           |   uint16
|       +-- target-kdf?
|           |   uint16
+-- server-authentication
|   +-- ca-certs! {server-auth-x509-cert}?
|       |   +---u ts:local-or-truststore-certs-grouping
+-- ee-certs! {server-auth-x509-cert}?
|       |   +---u ts:local-or-truststore-certs-grouping
+-- raw-public-keys! {server-auth-raw-public-key}?
|       |   +---u ts:local-or-truststore-public-keys-grouping
+-- tls12-psks?          empty {server-auth-tls12-psk}?
+-- tls13-epsks?        empty {server-auth-tls13-epsk}?
+-- hello-params {tlscmn:hello-params}?
|   +---u tlscmn:hello-params-grouping
+-- keepalives {tls-client-keepalives}?
+-- peer-allowed-to-send?    empty
+-- test-peer-aliveness!
|   +-- max-wait?          uint16
|   +-- max-attempts?     uint8

```

Comments:

- \* The "client-identity" node, which is optionally configured (as client authentication MAY occur at a higher protocol layer), configures identity credentials, each enabled by a "feature" statement defined in Section 3.1.1.
- \* The "server-authentication" node configures trust anchors for authenticating the TLS server, with each option enabled by a "feature" statement.
- \* The "hello-params" node, which must be enabled by a feature, configures parameters for the TLS sessions established by this configuration.
- \* The "keepalives" node, which must be enabled by a feature, configures a "presence" container for testing the aliveness of the TLS server. The aliveness-test occurs at the TLS protocol layer.
- \* For the referenced grouping statement(s):
  - The "local-or-keystore-end-entity-cert-with-key-grouping" grouping is discussed in Section 2.1.3.6 of [I-D.ietf-netconf-keystore].
  - The "local-or-keystore-asymmetric-key-grouping" grouping is discussed in Section 2.1.3.4 of [I-D.ietf-netconf-keystore].
  - The "local-or-keystore-symmetric-key-grouping" grouping is discussed in Section 2.1.3.3 of [I-D.ietf-netconf-keystore].
  - The "local-or-truststore-certs-grouping" grouping is discussed in Section 2.1.3.1 of [I-D.ietf-netconf-trust-anchors].
  - The "local-or-truststore-public-keys-grouping" grouping is discussed in Section 2.1.3.2 of [I-D.ietf-netconf-trust-anchors].
  - The "hello-params-grouping" grouping is discussed in Section 2.1.3.1 in this document.

### 3.1.3. Protocol-accessible Nodes

The "ietf-tls-client" module defines only "grouping" statements that are used by other modules to instantiate protocol-accessible nodes.

### 3.2. Example Usage

This section presents two examples showing the "tls-client-grouping" grouping populated with some data. These examples are effectively the same except the first configures the client identity using a local key while the second uses a key configured in a keystore. Both examples are consistent with the examples presented in Section 2 of [I-D.ietf-netconf-trust-anchors] and Section 3.2 of [I-D.ietf-netconf-keystore].



The following configuration example uses local-definitions for the client identity and server authentication:

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

<!-- The outermost element below doesn't exist in the data model. -->  
 <!-- It simulates if the "grouping" were a "container" instead. -->

```
<tls-client
  xmlns="urn:ietf:params:xml:ns:yang:ietf-tls-client"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
  <!-- how this client will authenticate itself to the server -->
  <client-identity>
    <certificate>
      <local-definition>
        <public-key-format>ct:subject-public-key-info-format\
</public-key-format>
        <public-key>BASE64VALUE=</public-key>
        <private-key-format>ct:rsa-private-key-format</priva\
te-key-format>
        <cleartext-private-key>BASE64VALUE=</cleartext-priva\
te-key>
        <cert-data>BASE64VALUE=</cert-data>
      </local-definition>
    </certificate>
    <!-- TESTED, BUT COMMENTED OUT DUE TO ONLY ONE ALLOWED AT A \
TIME
    <raw-private-key>
      <local-definition>
        <public-key-format>ct:subject-public-key-info-format</pu\
blic-key-format>
        <public-key>BASE64VALUE=</public-key>
        <private-key-format>ct:rsa-private-key-format</private-k\
ey-format>
        <cleartext-private-key>BASE64VALUE=</cleartext-private-k\
ey>
      </local-definition>
    </raw-private-key>
    -->
    <!-- USE ONLY ONE AT A TIME
    <tls12-psk>
      <local-definition>
        <key-format>ct:octet-string-key-format</key-format>
        <cleartext-key>BASE64VALUE=</cleartext-key>
      </local-definition>
      <id>example_id_string</id>
    </tls12-psk>
    -->
```

```

    <tls13-epsk>
      <local-definition>
        <key-format>ct:octet-string-key-format</key-format>
        <cleartext-key>BASE64VALUE=</cleartext-key>
      </local-definition>
      <external-identity>example_external_id</external-identity>
y>
      <hash>sha-256</hash>
      <context>example_context_string</context>
      <target-protocol>8443</target-protocol>
      <target-kdf>12345</target-kdf>
    </tls13-epsk>
  </client-identity>
  <!-- which certificates will this client trust -->
  <server-authentication>
    <ca-certs>
      <local-definition>
        <certificate>
          <name>Server Cert Issuer #1</name>
          <cert-data>BASE64VALUE=</cert-data>
        </certificate>
        <certificate>
          <name>Server Cert Issuer #2</name>
          <cert-data>BASE64VALUE=</cert-data>
        </certificate>
      </local-definition>
    </ca-certs>
    <ee-certs>
      <local-definition>
        <certificate>
          <name>My Application #1</name>
          <cert-data>BASE64VALUE=</cert-data>
        </certificate>
        <certificate>
          <name>My Application #2</name>
          <cert-data>BASE64VALUE=</cert-data>
        </certificate>
      </local-definition>
    </ee-certs>
    <raw-public-keys>
      <local-definition>
        <public-key>
          <name>corp-fw1</name>
          <public-key-format>ct:subject-public-key-info-fo\
rmat</public-key-format>
          <public-key>BASE64VALUE=</public-key>
        </public-key>
        <public-key>

```

```

        <name>corp-fw1</name>
        <public-key-format>ct:subject-public-key-info-fo\
rmat</public-key-format>
        <public-key>BASE64VALUE=</public-key>
      </public-key>
    </local-definition>
  </raw-public-keys>
  <tls12-psks/>
  <tls13-epsks/>
</server-authentication>
<keepalives>
  <test-peer-aliveness>
    <max-wait>30</max-wait>
    <max-attempts>3</max-attempts>
  </test-peer-aliveness>
</keepalives>
</tls-client>

```

The following configuration example uses keystore-references for the client identity and truststore-references for server authentication: from the keystore:

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

```

<!-- The outermost element below doesn't exist in the data model. -->
<!-- It simulates if the "grouping" were a "container" instead. -->

<tls-client xmlns="urn:ietf:params:xml:ns:yang:ietf-tls-client">
  <!-- how this client will authenticate itself to the server -->
  <client-identity>
    <certificate>
      <keystore-reference>
        <asymmetric-key>rsa-asymmetric-key</asymmetric-key>
        <certificate>ex-rsa-cert</certificate>
      </keystore-reference>
    </certificate>
    <!-- TESTED, BUT COMMENTED OUT DUE TO ONLY ONE ALLOWED AT A \
TIME
    <raw-private-key>
      <keystore-reference>raw-private-key</keystore-reference>
    </raw-private-key>
    -->
    <!-- USE ONLY ONE AT A TIME
    <tls12-psk>
      <keystore-reference>encrypted-symmetric-key</keystore-re\
ference>
      <id>example_id_string</id>
    </tls12-psk>

```

```

-->
<tls13-epsk>
  <keystore-reference>encrypted-symmetric-key</keystore-re\
ference>
  <external-identity>example_external_id</external-identit\
y>
  <hash>sha-256</hash>
  <context>example_context_string</context>
  <target-protocol>8443</target-protocol>
  <target-kdf>12345</target-kdf>
</tls13-epsk>
</client-identity>
<!-- which certificates will this client trust -->
<server-authentication>
  <ca-certs>
    <truststore-reference>trusted-server-ca-certs</truststor\
e-reference>
  </ca-certs>
  <ee-certs>
    <truststore-reference>trusted-server-ee-certs</truststor\
e-reference>
  </ee-certs>
  <raw-public-keys>
    <truststore-reference>Raw Public Keys for TLS Servers</t\
ruststore-reference>
  </raw-public-keys>
  <tls12-psks/>
  <tls13-epsks/>
</server-authentication>
<keepalives>
  <test-peer-aliveness>
    <max-wait>30</max-wait>
    <max-attempts>3</max-attempts>
  </test-peer-aliveness>
</keepalives>
</tls-client>

```

### 3.3. YANG Module

This YANG module has normative references to [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore], and Informative references to [RFC5246], [RFC8446], [I-D.ietf-tls-external-psk-importer] and [I-D.ietf-tls-external-psk-guidance].

```
<CODE BEGINS> file "ietf-tls-client@2022-03-07.yang"
```

```
module ietf-tls-client {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-tls-client";
  prefix tlsc;

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

  import ietf-crypto-types {
    prefix ct;
    reference
      "RFC AAAA: YANG Data Types and Groupings for Cryptography";
  }

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

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

  import ietf-tls-common {
    prefix tlscmn;
    revision-date 2022-03-07; // stable grouping definitions
    reference
      "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
  }

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

  contact
    "WG List:  NETCONF WG list <mailto:netconf@ietf.org>
    WG Web:    https://datatracker.ietf.org/wg/netconf
    Author:    Kent Watsen <mailto:kent+ietf@watsen.net>
    Author:    Gary Wu <mailto:garywu@cisco.com>
    Author:    Jeff Hartley <mailto:jeff.hartley@commscope.com>";

  description
    "This module defines reusable groupings for TLS clients that
```

can be used as a basis for specific TLS client instances.

Copyright (c) 2021 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

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

```
revision 2022-03-07 {
  description
    "Initial version";
  reference
    "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
}

// Features

feature tls-client-keepalives {
  description
    "Per socket TLS keepalive parameters are configurable for
    TLS clients on the server implementing this feature.";
}

feature client-ident-x509-cert {
  description
    "Indicates that the client supports identifying itself
    using X.509 certificates.";
  reference
    "RFC 5280:
    Internet X.509 Public Key Infrastructure Certificate
    and Certificate Revocation List (CRL) Profile";
}
```

```
feature client-ident-raw-public-key {
  description
    "Indicates that the client supports identifying itself
    using raw public keys.";
  reference
    "RFC 7250:
    Using Raw Public Keys in Transport Layer Security (TLS)
    and Datagram Transport Layer Security (DTLS)";
}

feature client-ident-tls12-psk {
  description
    "Indicates that the client supports identifying itself
    using TLS-1.2 PSKs (pre-shared or pairwise-symmetric keys).";
  reference
    "RFC 4279:
    Pre-Shared Key Ciphersuites for Transport Layer Security
    (TLS)";
}

feature client-ident-tls13-epsk {
  description
    "Indicates that the client supports identifying itself
    using TLS-1.3 External PSKs (pre-shared keys).";
  reference
    "RFC 8446:
    The Transport Layer Security (TLS) Protocol Version 1.3";
}

feature server-auth-x509-cert {
  description
    "Indicates that the client supports authenticating servers
    using X.509 certificates.";
  reference
    "RFC 5280:
    Internet X.509 Public Key Infrastructure Certificate
    and Certificate Revocation List (CRL) Profile";
}

feature server-auth-raw-public-key {
  description
    "Indicates that the client supports authenticating servers
    using raw public keys.";
  reference
    "RFC 7250:
    Using Raw Public Keys in Transport Layer Security (TLS)
    and Datagram Transport Layer Security (DTLS)";
}
```

```
feature server-auth-tls12-psk {
  description
    "Indicates that the client supports authenticating servers
    using PSKs (pre-shared or pairwise-symmetric keys).";
  reference
    "RFC 4279:
    Pre-Shared Key Ciphersuites for Transport Layer Security
    (TLS)";
}

feature server-auth-tls13-epsk {
  description
    "Indicates that the client supports authenticating servers
    using TLS-1.3 External PSKs (pre-shared keys).";
  reference
    "RFC 8446:
    The Transport Layer Security (TLS) Protocol Version 1.3";
}

// Groupings

grouping tls-client-grouping {
  description
    "A reusable grouping for configuring a TLS client without
    any consideration for how an underlying TCP session is
    established.

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

  container client-identity {
    nacm:default-deny-write;
    presence
      "Indicates that a TLS-level client identity has been
      configured. This statement is present so the mandatory
      descendant do not imply that this node must be configured.";
    description
      "Identity credentials the TLS client MAY present when
      establishing a connection to a TLS server. If not
      configured, then client authentication is presumed to
      occur a protocol layer above TLS. When configured,
      and requested by the TLS server when establishing a
```



```

    TLS session, these credentials are passed in the
    Certificate message defined in Section 7.4.2 of
    RFC 5246 and Section 4.4.2 in RFC 8446.";
reference
  "RFC 5246: The Transport Layer Security (TLS)
    Protocol Version 1.2
  RFC 8446: The Transport Layer Security (TLS)
    Protocol Version 1.3
  RFC CCCC: A YANG Data Model for a Keystore";
choice auth-type {
  mandatory true;
  description
    "A choice amongst authentication types, of which one must
    be enabled (via its associated 'feature') and selected.";
  case certificate {
    if-feature "client-ident-x509-cert";
    container certificate {
      description
        "Specifies the client identity using a certificate.";
      uses
        ks:local-or-keystore-end-entity-cert-with-key-grouping{
          refine "local-or-keystore/local/local-definition" {
            must 'public-key-format'
              + ' = "ct:subject-public-key-info-format"';
          }
          refine "local-or-keystore/keystore/keystore-reference"
            + "/asymmetric-key" {
            must 'deref(..)/ks:public-key-format'
              + ' = "ct:subject-public-key-info-format"';
          }
        }
      }
  }
  case raw-public-key {
    if-feature "client-ident-raw-public-key";
    container raw-private-key {
      description
        "Specifies the client identity using a raw
        private key.";
      uses ks:local-or-keystore-asymmetric-key-grouping {
        refine "local-or-keystore/local/local-definition" {
          must 'public-key-format'
            + ' = "ct:subject-public-key-info-format"';
        }
        refine "local-or-keystore/keystore"
          + "/keystore-reference" {
          must 'deref(..)/ks:public-key-format'
            + ' = "ct:subject-public-key-info-format"';
        }
      }
    }
  }
}
```

```
    }
  }
}
case tls12-psk {
  if-feature "client-ident-tls12-psk";
  container tls12-psk {
    description
      "Specifies the client identity using a PSK (pre-shared
       or pairwise-symmetric key).";
    uses ks:local-or-keystore-symmetric-key-grouping;
    leaf id {
      type string;
      description
        "The key 'psk_identity' value used in the TLS
         'ClientKeyExchange' message.";
      reference
        "RFC 4279: Pre-Shared Key Ciphersuites for
         Transport Layer Security (TLS)";
    }
  }
}
case tls13-epsk {
  if-feature "client-ident-tls13-epsk";
  container tls13-epsk {
    description
      "An External Pre-Shared Key (EPSK) is established
       or provisioned out-of-band, i.e., not from a TLS
       connection. An EPSK is a tuple of (Base Key,
       External Identity, Hash). External PSKs MUST NOT
       be imported for (D)TLS 1.2 or prior versions. When
       PSKs are provisioned out of band, the PSK identity
       and the KDF hash algorithm to be used with the PSK
       MUST also be provisioned.

       The structure of this container is designed
       to satisfy the requirements of RFC 8446
       Section 4.2.11, the recommendations from I-D
       ietf-tls-external-psk-guidance Section 6,
       and the EPSK input fields detailed in I-D
       draft-ietf-tls-external-psk-importer
       Section 3.1. The base-key is based upon
       ks:local-or-keystore-symmetric-key-grouping
       in order to provide users with flexible and
       secure storage options.";
    reference
      "RFC 8446: The Transport Layer Security (TLS)
       Protocol Version 1.3"
```

```
I-D.ietf-tls-external-psk-importer:
    Importing External PSKs for TLS
I-D.ietf-tls-external-psk-guidance:
    Guidance for External PSK Usage in TLS";
uses ks:local-or-keystore-symmetric-key-grouping;
leaf external-identity {
    type string;
    mandatory true;
    description
        "As per Section 4.2.11 of RFC 8446, and Section 4.1
        of I-D. ietf-tls-external-psk-guidance:
        A sequence of bytes used to identify an EPSK. A
        label for a pre-shared key established externally.";
    reference
        "RFC 8446: The Transport Layer Security (TLS)
        Protocol Version 1.3
        I-D.ietf-tls-external-psk-guidance:
        Guidance for External PSK Usage in TLS";
}
leaf hash {
    type tlscmn:epsk-supported-hash;
    mandatory true;
    description
        "As per Section 4.2.11 of RFC 8446, for externally
        established PSKs, the Hash algorithm MUST be set
        when the PSK is established or default to SHA-256
        if no such algorithm is defined. The server MUST
        ensure that it selects a compatible PSK (if any)
        and cipher suite. Each PSK MUST only be used with
        a single hash function.";
    reference
        "RFC 8446: The Transport Layer Security (TLS)
        Protocol Version 1.3";
}
leaf context {
    type string;
    description
        "As per Section 4.1 of I-D.
        ietf-tls-external-psk-guidance: Context may include
        information about peer roles or identities to
        mitigate Selfie-style reflection attacks [Selfie].
        If the EPSK is a key derived from some other
        protocol or sequence of protocols, context
        MUST include a channel binding for the deriving
        protocols [RFC5056]. The details of this binding
        are protocol specific.";
    reference
        "I-D.ietf-tls-external-psk-importer:
```

```

        Importing External PSKs for TLS
        I-D.ietf-tls-external-psk-guidance:
        Guidance for External PSK Usage in TLS";
    }
    leaf target-protocol {
        type uint16;
        description
            "As per Section 3.1 of I-D.
            ietf-tls-external-psk-guidance:
            The protocol for which a PSK is imported for use.";
        reference
            "I-D.ietf-tls-external-psk-importer:
            Importing External PSKs for TLS";
    }
    leaf target-kdf {
        type uint16;
        description
            "As per Section 3.1 of I-D.
            ietf-tls-external-psk-guidance:
            The specific Key Derivation Function (KDF) for which
            a PSK is imported for use.";
        reference
            "I-D.ietf-tls-external-psk-importer:
            Importing External PSKs for TLS";
    }
}
}
} // container client-identity

container server-authentication {
    nacm:default-deny-write;
    must 'ca-certs or ee-certs or raw-public-keys or tls12-psks
    or tls13-epsks';
    description
        "Specifies how the TLS client can authenticate TLS servers.
        Any combination of credentials is additive and unordered.

        Note that no configuration is required for PSK (pre-shared
        or pairwise-symmetric key) based authentication as the key
        is necessarily the same as configured in the '../client-
        identity' node.";
    container ca-certs {
        if-feature "server-auth-x509-cert";
        presence
            "Indicates that CA certificates have been configured.
            This statement is present so the mandatory descendant
            nodes do not imply that this node must be configured.";
    }
}

```

```
description
  "A set of certificate authority (CA) certificates used by
  the TLS client to authenticate TLS server certificates.
  A server certificate is authenticated if it has a valid
  chain of trust to a configured CA certificate.";
reference
  "RFC BBBB: A YANG Data Model for a Truststore";
uses ts:local-or-truststore-certs-grouping;
}
container ee-certs {
  if-feature "server-auth-x509-cert";
  presence
    "Indicates that EE certificates have been configured.
    This statement is present so the mandatory descendant
    nodes do not imply that this node must be configured.";
  description
    "A set of server certificates (i.e., end entity
    certificates) used by the TLS client to authenticate
    certificates presented by TLS servers. A server
    certificate is authenticated if it is an exact
    match to a configured server certificate.";
  reference
    "RFC BBBB: A YANG Data Model for a Truststore";
  uses ts:local-or-truststore-certs-grouping;
}
container raw-public-keys {
  if-feature "server-auth-raw-public-key";
  presence
    "Indicates that raw public keys have been configured.
    This statement is present so the mandatory descendant
    nodes do not imply that this node must be configured.";
  description
    "A set of raw public keys used by the TLS client to
    authenticate raw public keys presented by the TLS
    server. A raw public key is authenticated if it
    is an exact match to a configured raw public key.";
  reference
    "RFC BBBB: A YANG Data Model for a Truststore";
  uses ts:local-or-truststore-public-keys-grouping {
    refine "local-or-truststore/local/local-definition"
      + "/public-key" {
        must 'public-key-format'
        + ' = "ct:subject-public-key-info-format"';
      }
    refine "local-or-truststore/truststore"
      + "/truststore-reference" {
        must 'deref(.)/*/*ts:public-key-format'
        + ' = "ct:subject-public-key-info-format"';
      }
  }
}
```

```
    }
  }
}
leaf tls12-psks {
  if-feature "server-auth-tls12-psk";
  type empty;
  description
    "Indicates that the TLS client can authenticate TLS servers
    using configure PSKs (pre-shared or pairwise-symmetric
    keys).

    No configuration is required since the PSK value is the
    same as PSK value configured in the 'client-identity'
    node.";
}
leaf tls13-epsks {
  if-feature "server-auth-tls13-epsk";
  type empty;
  description
    "Indicates that the TLS client can authenticate TLS servers
    using configured external PSKs (pre-shared keys).

    No configuration is required since the PSK value is the
    same as PSK value configured in the 'client-identity'
    node.";
}
} // container server-authentication

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

container keepalives {
  nacm:default-deny-write;
  if-feature "tls-client-keepalives";
  description
    "Configures the keepalive policy for the TLS client.";
  leaf peer-allowed-to-send {
    type empty;
    description
      "Indicates that the remote TLS server is allowed to send
      HeartbeatRequest messages, as defined by RFC 6520
      to this TLS client.";
    reference

```

```
        "RFC 6520: Transport Layer Security (TLS) and Datagram
        Transport Layer Security (DTLS) Heartbeat Extension";
    }
    container test-peer-aliveness {
        presence
            "Indicates that the TLS client proactively tests the
            aliveness of the remote TLS server.";
        description
            "Configures the keep-alive policy to proactively test
            the aliveness of the TLS server.  An unresponsive
            TLS server is dropped after approximately max-wait
            * max-attempts seconds.  The TLS client MUST send
            HeartbeatRequest messages, as defined by RFC 6520.";
        reference
            "RFC 6520: Transport Layer Security (TLS) and Datagram
            Transport Layer Security (DTLS) Heartbeat Extension";
        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.";
        }
    }
}
} // grouping tls-client-grouping
}

<CODE ENDS>
```

#### 4. The "ietf-tls-server" Module

This section defines a YANG 1.1 module called "ietf-tls-server". A high-level overview of the module is provided in Section 4.1. Examples illustrating the module's use are provided in Examples (Section 4.2). The YANG module itself is defined in Section 4.3.

##### 4.1. Data Model Overview

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

###### 4.1.1. Features

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

Features:

```
+-- tls-server-keepalives
+-- server-ident-x509-cert
+-- server-ident-raw-public-key
+-- server-ident-psk
+-- client-auth-supported
+-- client-auth-x509-cert
+-- client-auth-raw-public-key
+-- client-auth-psk
```

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

###### 4.1.2. Groupings

The "ietf-tls-server" module defines the following "grouping" statement:

```
* tls-server-grouping
```

This grouping is presented in the following subsection.

###### 4.1.2.1. The "tls-server-grouping" Grouping

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



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

```

grouping tls-server-grouping
+-- server-identity
|   +-- (auth-type)
|       +--:(certificate) {server-ident-x509-cert}?
|           +-- certificate
|               +---u ks:local-or-keystore-end-entity-cert-with-key-\
grouping
|       +--:(raw-private-key) {server-ident-raw-public-key}?
|           +-- raw-private-key
|               +---u ks:local-or-keystore-asymmetric-key-grouping
+--:(tls12-psk) {server-ident-tls12-psk}?
|   +-- tls12-psk
|       +---u ks:local-or-keystore-symmetric-key-grouping
|       +-- id_hint?
|           string
+--:(tls13-epsk) {server-ident-tls13-epsk}?
|   +-- tls13-epsk
|       +---u ks:local-or-keystore-symmetric-key-grouping
|       +-- external-identity
|           |   string
|       +-- hash
|           |   tlscmn:epsk-supported-hash
|       +-- context?
|           |   string
|       +-- target-protocol?
|           |   uint16
|       +-- target-kdf?
|           |   uint16
+-- client-authentication! {client-auth-supported}?
|   +-- ca-certs! {client-auth-x509-cert}?
|       |   +---u ts:local-or-truststore-certs-grouping
+-- ee-certs! {client-auth-x509-cert}?
|       |   +---u ts:local-or-truststore-certs-grouping
+-- raw-public-keys! {client-auth-raw-public-key}?
|       |   +---u ts:local-or-truststore-public-keys-grouping
+-- tls12-psks?          empty {client-auth-tls12-psk}?
+-- tls13-epsks?        empty {client-auth-tls13-epsk}?
+-- hello-params {tlscmn:hello-params}?
|   +---u tlscmn:hello-params-grouping
+-- keepalives {tls-server-keepalives}?
+-- peer-allowed-to-send?    empty
+-- test-peer-aliveness!
|   +-- max-wait?          uint16
|   +-- max-attempts?      uint8

```

Comments:

- \* The "server-identity" node configures identity credentials, each of which is enabled by a "feature".
- \* The "client-authentication" node, which is optionally configured (as client authentication MAY occur at a higher protocol layer), configures trust anchors for authenticating the TLS client, with each option enabled by a "feature" statement.
- \* The "hello-params" node, which must be enabled by a feature, configures parameters for the TLS sessions established by this configuration.
- \* The "keepalives" node, which must be enabled by a feature, configures a flag enabling the TLS client to test the aliveness of the TLS server, as well as a "presence" container for testing the aliveness of the TLS client. The aliveness-tests occurs at the TLS protocol layer.
- \* For the referenced grouping statement(s):
  - The "local-or-keystore-end-entity-cert-with-key-grouping" grouping is discussed in Section 2.1.3.6 of [I-D.ietf-netconf-keystore].
  - The "local-or-keystore-asymmetric-key-grouping" grouping is discussed in Section 2.1.3.4 of [I-D.ietf-netconf-keystore].
  - The "local-or-keystore-symmetric-key-grouping" grouping is discussed in Section 2.1.3.3 of [I-D.ietf-netconf-keystore].
  - The "local-or-truststore-public-keys-grouping" grouping is discussed in Section 2.1.3.2 of [I-D.ietf-netconf-trust-anchors].
  - The "local-or-truststore-certs-grouping" grouping is discussed in Section 2.1.3.1 of [I-D.ietf-netconf-trust-anchors].
  - The "hello-params-grouping" grouping is discussed in Section 2.1.3.1 in this document.

#### 4.1.3. Protocol-accessible Nodes

The "ietf-tls-server" module defines only "grouping" statements that are used by other modules to instantiate protocol-accessible nodes.

#### 4.2. Example Usage

This section presents two examples showing the "tls-server-grouping" grouping populated with some data. These examples are effectively the same except the first configures the server identity using a local key while the second uses a key configured in a keystore. Both examples are consistent with the examples presented in Section 2 of [I-D.ietf-netconf-trust-anchors] and Section 3.2 of

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

The following configuration example uses local-definitions for the server identity and client authentication:

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

<!-- The outermost element below doesn't exist in the data model. -->  
 <!-- It simulates if the "grouping" were a "container" instead. -->

```
<tls-server
  xmlns="urn:ietf:params:xml:ns:yang:ietf-tls-server"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
  <!-- how this server will authenticate itself to the client -->
  <server-identity>
    <certificate>
      <local-definition>
        <public-key-format>ct:subject-public-key-info-format\
</public-key-format>
        <public-key>BASE64VALUE=</public-key>
        <private-key-format>ct:rsa-private-key-format</priva\
te-key-format>
        <cleartext-private-key>BASE64VALUE=</cleartext-priva\
te-key>
        <cert-data>BASE64VALUE=</cert-data>
      </local-definition>
    </certificate>
    <!-- TESTED, BUT COMMENTED OUT DUE TO ONLY ONE ALLOWED AT A \
TIME
    <raw-private-key>
      <local-definition>
        <public-key-format>ct:subject-public-key-info-format</pu\
blic-key-format>
        <public-key>BASE64VALUE=</public-key>
        <private-key-format>ct:rsa-private-key-format</private-k\
ey-format>
        <cleartext-private-key>BASE64VALUE=</cleartext-private-k\
ey>
      </local-definition>
    </raw-private-key>
    -->
    <!-- USE ONLY ONE AT A TIME
  <tls12-psk>
    <local-definition>
      <key-format>ct:octet-string-key-format</key-format>
      <cleartext-key>BASE64VALUE=</cleartext-key>
    </local-definition>
    <id_hint>example_id_hint</id_hint>
```

```

    </tls12-psk>
    -->
    <tls13-epsk>
      <local-definition>
        <key-format>ct:octet-string-key-format</key-format>
        <cleartext-key>BASE64VALUE=</cleartext-key>
      </local-definition>
      <external-identity>example_external_id</external-identity>
y>
      <hash>sha-256</hash>
      <context>example_context_string</context>
      <target-protocol>8443</target-protocol>
      <target-kdf>12345</target-kdf>
    </tls13-epsk>
  </server-identity>
  <!-- which certificates will this server trust -->
  <client-authentication>
    <ca-certs>
      <local-definition>
        <certificate>
          <name>Identity Cert Issuer #1</name>
          <cert-data>BASE64VALUE=</cert-data>
        </certificate>
        <certificate>
          <name>Identity Cert Issuer #2</name>
          <cert-data>BASE64VALUE=</cert-data>
        </certificate>
      </local-definition>
    </ca-certs>
    <ee-certs>
      <local-definition>
        <certificate>
          <name>Application #1</name>
          <cert-data>BASE64VALUE=</cert-data>
        </certificate>
        <certificate>
          <name>Application #2</name>
          <cert-data>BASE64VALUE=</cert-data>
        </certificate>
      </local-definition>
    </ee-certs>
    <raw-public-keys>
      <local-definition>
        <public-key>
          <name>User A</name>
          <public-key-format>ct:subject-public-key-info-format</public-key-format>
          <public-key>BASE64VALUE=</public-key>

```

```

        </public-key>
        <public-key>
            <name>User B</name>
            <public-key-format>ct:subject-public-key-info-fo\
rmat</public-key-format>
            <public-key>BASE64VALUE=</public-key>
        </public-key>
    </local-definition>
</raw-public-keys>
<tls12-psks/>
<tls13-epsks/>
</client-authentication>
<keepalives>
    <peer-allowed-to-send/>
</keepalives>
</tls-server>

```

The following configuration example uses keystore-references for the server identity and truststore-references for client authentication: from the keystore:

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

```

<!-- The outermost element below doesn't exist in the data model. -->
<!-- It simulates if the "grouping" were a "container" instead. -->

<tls-server xmlns="urn:ietf:params:xml:ns:yang:ietf-tls-server">
    <!-- how this server will authenticate itself to the client -->
    <server-identity>
        <certificate>
            <keystore-reference>
                <asymmetric-key>rsa-asymmetric-key</asymmetric-key>
                <certificate>ex-rsa-cert</certificate>
            </keystore-reference>
        </certificate>
        <!-- TESTED, BUT COMMENTED OUT DUE TO ONLY ONE ALLOWED AT A \
TIME
-->
        <!-- USE ONLY ONE AT A TIME
        <raw-private-key>
            <keystore-reference>raw-private-key</keystore-reference>
        </raw-private-key>
        -->
        <!-- USE ONLY ONE AT A TIME
        <tls12-psk>
            <keystore-reference>encrypted-symmetric-key</keystore-re\
ference>
            <id_hint>example_id_hint</id_hint>

```

```

        </tls12-psk>
        -->
        <tls13-epsk>
            <keystore-reference>encrypted-symmetric-key</keystore-re\
ference>
            <external-identity>example_external_id</external-identit\
y>
            <hash>sha-256</hash>
            <context>example_context_string</context>
            <target-protocol>8443</target-protocol>
            <target-kdf>12345</target-kdf>
        </tls13-epsk>
    </server-identity>
    <!-- which certificates will this server trust -->
    <client-authentication>
        <ca-certs>
            <truststore-reference>trusted-client-ca-certs</truststor\
e-reference>
        </ca-certs>
        <ee-certs>
            <truststore-reference>trusted-client-ee-certs</truststor\
e-reference>
        </ee-certs>
        <raw-public-keys>
            <truststore-reference>Raw Public Keys for TLS Clients</t\
ruststore-reference>
        </raw-public-keys>
        <tls12-psks/>
        <tls13-epsks/>
    </client-authentication>
    <keepalives>
        <peer-allowed-to-send/>
    </keepalives>
</tls-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 [RFC5246], [RFC8446], [I-D.ietf-tls-external-psk-importer] and [I-D.ietf-tls-external-psk-guidance].

```
<CODE BEGINS> file "ietf-tls-server@2022-03-07.yang"
```

```
module ietf-tls-server {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-tls-server";
  prefix tlss;

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

  import ietf-crypto-types {
    prefix ct;
    reference
      "RFC AAAA: YANG Data Types and Groupings for Cryptography";
  }

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

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

  import ietf-tls-common {
    prefix tlscmn;
    revision-date 2022-03-07; // stable grouping definitions
    reference
      "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
  }

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

  contact
    "WG List:  NETCONF WG list <mailto:netconf@ietf.org>
    WG Web:    https://datatracker.ietf.org/wg/netconf
    Author:    Kent Watsen <mailto:kent+ietf@watsen.net>
    Author:    Gary Wu <mailto:garywu@cisco.com>
    Author:    Jeff Hartley <mailto:jeff.hartley@commscope.com>";

  description
    "This module defines reusable groupings for TLS servers that
```

can be used as a basis for specific TLS server instances.

Copyright (c) 2021 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

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

```
revision 2022-03-07 {
  description
    "Initial version";
  reference
    "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
}

// Features

feature tls-server-keepalives {
  description
    "Per socket TLS keepalive parameters are configurable for
    TLS servers on the server implementing this feature.";
}

feature server-ident-x509-cert {
  description
    "Indicates that the server supports identifying itself
    using X.509 certificates.";
  reference
    "RFC 5280:
    Internet X.509 Public Key Infrastructure Certificate
    and Certificate Revocation List (CRL) Profile";
}
```



```
feature server-ident-raw-public-key {
  description
    "Indicates that the server supports identifying itself
    using raw public keys.";
  reference
    "RFC 7250:
    Using Raw Public Keys in Transport Layer Security (TLS)
    and Datagram Transport Layer Security (DTLS)";
}

feature server-ident-tls12-psk {
  description
    "Indicates that the server supports identifying itself
    using TLS-1.2 PSKs (pre-shared or pairwise-symmetric keys).";
  reference
    "RFC 4279:
    Pre-Shared Key Ciphersuites for Transport Layer Security
    (TLS)";
}

feature server-ident-tls13-epsk {
  description
    "Indicates that the server supports identifying itself
    using TLS-1.3 External PSKs (pre-shared keys).";
  reference
    "RFC 8446:
    The Transport Layer Security (TLS) Protocol Version 1.3";
}

feature client-auth-supported {
  description
    "Indicates that the configuration for how to authenticate
    clients can be configured herein. TLS-level client
    authentication may not be needed when client authentication
    is expected to occur only at another protocol layer.";
}

feature client-auth-x509-cert {
  description
    "Indicates that the server supports authenticating clients
    using X.509 certificates.";
  reference
    "RFC 5280:
    Internet X.509 Public Key Infrastructure Certificate
    and Certificate Revocation List (CRL) Profile";
}

feature client-auth-raw-public-key {
```

```
description
  "Indicates that the server supports authenticating clients
   using raw public keys.";
reference
  "RFC 7250:
   Using Raw Public Keys in Transport Layer Security (TLS)
   and Datagram Transport Layer Security (DTLS)";
}

feature client-auth-tls12-psk {
  description
    "Indicates that the server supports authenticating clients
     using PSKs (pre-shared or pairwise-symmetric keys).";
  reference
    "RFC 4279:
     Pre-Shared Key Ciphersuites for Transport Layer Security
     (TLS)";
}

feature client-auth-tls13-epsk {
  description
    "Indicates that the server supports authenticating clients
     using TLS-1.3 External PSKs (pre-shared keys).";
  reference
    "RFC 8446:
     The Transport Layer Security (TLS) Protocol Version 1.3";
}

// Groupings

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

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

  container server-identity {
    nacm:default-deny-write;
    description
```

```
"A locally-defined or referenced end-entity certificate,
including any configured intermediate certificates, the
TLS server will present when establishing a TLS connection
in its Certificate message, as defined in Section 7.4.2
in RFC 5246 and Section 4.4.2 in RFC 8446.";
reference
  "RFC 5246: The Transport Layer Security (TLS) Protocol
    Version 1.2
  RFC 8446: The Transport Layer Security (TLS) Protocol
    Version 1.3
  RFC CCCC: A YANG Data Model for a Keystore";
choice auth-type {
  mandatory true;
  description
    "A choice amongst authentication types, of which one must
    be enabled (via its associated 'feature') and selected.";
  case certificate {
    if-feature "server-ident-x509-cert";
    container certificate {
      description
        "Specifies the server identity using a certificate.";
      uses
        ks:local-or-keystore-end-entity-cert-with-key-grouping{
          refine "local-or-keystore/local/local-definition" {
            must 'public-key-format'
              + ' = "ct:subject-public-key-info-format"';
          }
          refine "local-or-keystore/keystore/keystore-reference"
            + "/asymmetric-key" {
            must 'deref(..)/../ks:public-key-format'
              + ' = "ct:subject-public-key-info-format"';
          }
        }
      }
    }
  case raw-private-key {
    if-feature "server-ident-raw-public-key";
    container raw-private-key {
      description
        "Specifies the server identity using a raw
        private key.";
      uses ks:local-or-keystore-asymmetric-key-grouping {
        refine "local-or-keystore/local/local-definition" {
          must 'public-key-format'
            + ' = "ct:subject-public-key-info-format"';
        }
        refine "local-or-keystore/keystore/keystore-reference"{
          must 'deref(..)/../ks:public-key-format'
```

```
        + ' = "ct:subject-public-key-info-format";
    }
}
}
case tls12-psk {
  if-feature "server-ident-tls12-psk";
  container tls12-psk {
    description
      "Specifies the server identity using a PSK (pre-shared
       or pairwise-symmetric key).";
    uses ks:local-or-keystore-symmetric-key-grouping;
    leaf id_hint {
      type string;
      description
        "The key 'psk_identity_hint' value used in the TLS
         'ServerKeyExchange' message.";
      reference
        "RFC 4279: Pre-Shared Key Ciphersuites for
         Transport Layer Security (TLS)";
    }
  }
}
case tls13-epsk {
  if-feature "server-ident-tls13-epsk";
  container tls13-epsk {
    description
      "An External Pre-Shared Key (EPSK) is established
       or provisioned out-of-band, i.e., not from a TLS
       connection. An EPSK is a tuple of (Base Key,
       External Identity, Hash). External PSKs MUST
       NOT be imported for (D)TLS 1.2 or prior versions.
       When PSKs are provisioned out of band, the PSK
       identity and the KDF hash algorithm to be used
       with the PSK MUST also be provisioned.

       The structure of this container is designed
       to satisfy the requirements of RFC 8446
       Section 4.2.11, the recommendations from
       I-D ietf-tls-external-psk-guidance Section 6,
       and the EPSK input fields detailed in
       I-D draft-ietf-tls-external-psk-importer
       Section 3.1. The base-key is based upon
       ks:local-or-keystore-symmetric-key-grouping
       in order to provide users with flexible and
       secure storage options.";
    reference
      "RFC 8446: The Transport Layer Security (TLS)";
```

```

        Protocol Version 1.3
    I-D.ietf-tls-external-psk-importer: Importing
        External PSKs for TLS
    I-D.ietf-tls-external-psk-guidance: Guidance
        for External PSK Usage in TLS";
uses ks:local-or-keystore-symmetric-key-grouping;
leaf external-identity {
    type string;
    mandatory true;
    description
        "As per Section 4.2.11 of RFC 8446, and Section 4.1
        of I-D. ietf-tls-external-psk-guidance: A sequence
        of bytes used to identify an EPSK. A label for a
        pre-shared key established externally.";
    reference
        "RFC 8446: The Transport Layer Security (TLS)
        Protocol Version 1.3
        I-D.ietf-tls-external-psk-guidance:
        Guidance for External PSK Usage in TLS";
}
leaf hash {
    type tlscmn:epsk-supported-hash;
    mandatory true;
    description
        "As per Section 4.2.11 of RFC 8446, for externally
        established PSKs, the Hash algorithm MUST be set
        when the PSK is established or default to SHA-256
        if no such algorithm is defined. The server MUST
        ensure that it selects a compatible PSK (if any)
        and cipher suite. Each PSK MUST only be used
        with a single hash function.";
    reference
        "RFC 8446: The Transport Layer Security (TLS)
        Protocol Version 1.3";
}
leaf context {
    type string;
    description
        "As per Section 4.1 of I-D.
        ietf-tls-external-psk-guidance: Context
        may include information about peer roles or
        identities to mitigate Selfie-style reflection
        attacks [Selfie]. If the EPSK is a key derived
        from some other protocol or sequence of protocols,
        context MUST include a channel binding for the
        deriving protocols [RFC5056]. The details of
        this binding are protocol specific.";
    reference

```

```

        "I-D.ietf-tls-external-psk-importer:
          Importing External PSKs for TLS
        I-D.ietf-tls-external-psk-guidance:
          Guidance for External PSK Usage in TLS";
    }
    leaf target-protocol {
        type uint16;
        description
            "As per Section 3.1 of I-D.
             ietf-tls-external-psk-guidance: The protocol
             for which a PSK is imported for use.";
        reference
            "I-D.ietf-tls-external-psk-importer:
             Importing External PSKs for TLS";
    }
    leaf target-kdf {
        type uint16;
        description
            "As per Section 3.1 of I-D.
             ietf-tls-external-psk-guidance: The specific Key
             Derivation Function (KDF) for which a PSK is
             imported for use.";
        reference
            "I-D.ietf-tls-external-psk-importer:
             Importing External PSKs for TLS";
    }
    }
}
} // container server-identity

container client-authentication {
    if-feature "client-auth-supported";
    nacm:default-deny-write;
    must 'ca-certs or ee-certs or raw-public-keys or tls12-psks
        or tls13-epsks';
    presence
        "Indicates that client authentication is supported (i.e.,
         that the server will request clients send certificates).
         If not configured, the TLS server SHOULD NOT request the
         TLS clients provide authentication credentials.";
    description
        "Specifies how the TLS server can authenticate TLS clients.
         Any combination of credentials is additive and unordered.

         Note that no configuration is required for PSK (pre-shared
         or pairwise-symmetric key) based authentication as the key
         is necessarily the same as configured in the '../server-
```

```
    identity' node.";
  container ca-certs {
    if-feature "client-auth-x509-cert";
    presence
      "Indicates that CA certificates have been configured.
       This statement is present so the mandatory descendant
       nodes do not imply that this node must be configured.";
    description
      "A set of certificate authority (CA) certificates used by
       the TLS server to authenticate TLS client certificates.
       A client certificate is authenticated if it has a valid
       chain of trust to a configured CA certificate.";
    reference
      "RFC BBBB: A YANG Data Model for a Truststore";
    uses ts:local-or-truststore-certs-grouping;
  }
  container ee-certs {
    if-feature "client-auth-x509-cert";
    presence
      "Indicates that EE certificates have been configured.
       This statement is present so the mandatory descendant
       nodes do not imply that this node must be configured.";
    description
      "A set of client certificates (i.e., end entity
       certificates) used by the TLS server to authenticate
       certificates presented by TLS clients. A client
       certificate is authenticated if it is an exact
       match to a configured client certificate.";
    reference
      "RFC BBBB: A YANG Data Model for a Truststore";
    uses ts:local-or-truststore-certs-grouping;
  }
  container raw-public-keys {
    if-feature "client-auth-raw-public-key";
    presence
      "Indicates that raw public keys have been configured.
       This statement is present so the mandatory descendant
       nodes do not imply that this node must be configured.";
    description
      "A set of raw public keys used by the TLS server to
       authenticate raw public keys presented by the TLS
       client. A raw public key is authenticated if it
       is an exact match to a configured raw public key.";
    reference
      "RFC BBBB: A YANG Data Model for a Truststore";
    uses ts:local-or-truststore-public-keys-grouping {
      refine "local-or-truststore/local/local-definition"
        + "/public-key" {
```

```
        must 'public-key-format'
        + ' = "ct:subject-public-key-info-format";
    }
    refine "local-or-truststore/truststore"
        + "/truststore-reference" {
        must 'deref(..)/../*/ts:public-key-format'
        + ' = "ct:subject-public-key-info-format";
    }
}
}
leaf tls12-psks {
    if-feature "client-auth-tls12-psk";
    type empty;
    description
        "Indicates that the TLS server can authenticate TLS clients
        using configured PSKs (pre-shared or pairwise-symmetric
        keys).

        No configuration is required since the PSK value is the
        same as PSK value configured in the 'server-identity'
        node.";
}
leaf tls13-epsks {
    if-feature "client-auth-tls13-epsk";
    type empty;
    description
        "Indicates that the TLS 1.3 server can authenticate TLS
        clients using configured external PSKs (pre-shared keys).

        No configuration is required since the PSK value is the
        same as PSK value configured in the 'server-identity'
        node.";
}
} // container client-authentication

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

container keepalives {
    nacm:default-deny-write;
    if-feature "tls-server-keepalives";
    description
        "Configures the keepalive policy for the TLS server.";
```



```
leaf peer-allowed-to-send {
  type empty;
  description
    "Indicates that the remote TLS client is allowed to send
    HeartbeatRequest messages, as defined by RFC 6520
    to this TLS server.";
  reference
    "RFC 6520: Transport Layer Security (TLS) and Datagram
    Transport Layer Security (DTLS) Heartbeat Extension";
}
container test-peer-aliveness {
  presence
    "Indicates that the TLS server proactively tests the
    aliveness of the remote TLS client.";
  description
    "Configures the keep-alive policy to proactively test
    the aliveness of the TLS client. An unresponsive
    TLS client is dropped after approximately max-wait
    * max-attempts seconds.";
  leaf max-wait {
    type uint16 {
      range "1..max";
    }
    units "seconds";
    default "30";
    description
      "Sets the amount of time in seconds after which if
      no data has been received from the TLS client, a
      TLS-level message will be sent to test the
      aliveness of the TLS client.";
  }
  leaf max-attempts {
    type uint8;
    default "3";
    description
      "Sets the maximum number of sequential keep-alive
      messages that can fail to obtain a response from
      the TLS client before assuming the TLS client is
      no longer alive.";
  }
}
} // container keepalives
} // grouping tls-server-grouping

}

<CODE ENDS>
```

## 5. Security Considerations

### 5.1. The "iana-tls-cipher-suite-algs" Module

The "iana-tls-cipher-suite-algs" YANG module defines a data model that 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.

This YANG module defines YANG identities, for a public IANA-maintained registry, and a single protocol-accessible read-only node for the subset of those identities supported by a server.

YANG identities are not security-sensitive, as they are statically defined in the publicly-accessible YANG module.

The protocol-accessible read-only node for the algorithms supported by a server is mildly sensitive, but not to the extent that special NACM annotations are needed to prevent read-access to regular authenticated administrators.

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

### 5.2. The "ietf-tls-common" YANG Module

The "ietf-tls-common" YANG module defines "grouping" statements that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

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

Since the module in this document only define groupings, these considerations are primarily for the designers of other modules that use these groupings.

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

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

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

### 5.3. The "ietf-tls-client" YANG Module

The "ietf-tls-client" YANG module defines "grouping" statements that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

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

Since the module in this document only define groupings, these considerations are primarily for the designers of other modules that use these groupings.

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

Please be aware that this module uses the "key" and "private-key" nodes from the "ietf-crypto-types" module [I-D.ietf-netconf-crypto-types], where said nodes have the NACM extension "default-deny-all" set, thus preventing unrestricted read-access to the cleartext key values.

All the writable data nodes defined by this module may be considered sensitive or vulnerable in some network environments. For instance, any modification to a key or reference to a key may dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been set for all data nodes defined in this module.

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

#### 5.4. The "ietf-tls-server" YANG Module

The "ietf-tls-server" YANG module defines "grouping" statements that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

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

Since the module in this document only define groupings, these considerations are primarily for the designers of other modules that use these groupings.

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

Please be aware that this module uses the "key" and "private-key" nodes from the "ietf-crypto-types" module [I-D.ietf-netconf-crypto-types], where said nodes have the NACM extension "default-deny-all" set, thus preventing unrestricted read-access to the cleartext key values.

All the writable data nodes defined by this module may be considered sensitive or vulnerable in some network environments. For instance, any modification to a key or reference to a key may dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been set for all data nodes defined in this module.

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

### 6. IANA Considerations

#### 6.1. The "IETF XML" Registry

This document registers four 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:iana-tls-cipher-suite-algs

Registrant Contact: IANA

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

URI: urn:ietf:params:xml:ns:yang:ietf-tls-common

Registrant Contact: The IESG

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

URI: urn:ietf:params:xml:ns:yang:ietf-tls-client

Registrant Contact: The IESG

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

URI: urn:ietf:params:xml:ns:yang:ietf-tls-server

Registrant Contact: The IESG

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

## 6.2. The "YANG Module Names" Registry

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

name: iana-tls-cipher-suite-algs  
namespace: urn:ietf:params:xml:ns:yang:iana-tls-cipher-suite-algs  
prefix: tlscsa  
reference: RFC FFFF

name: ietf-tls-common  
namespace: urn:ietf:params:xml:ns:yang:ietf-tls-common  
prefix: tlscmn  
reference: RFC FFFF

name: ietf-tls-client  
namespace: urn:ietf:params:xml:ns:yang:ietf-tls-client  
prefix: tlsc  
reference: RFC FFFF

name: ietf-tls-server  
namespace: urn:ietf:params:xml:ns:yang:ietf-tls-server  
prefix: tlss  
reference: RFC FFFF

## 6.3. The "iana-tls-cipher-suite-algs" Module

IANA is requested to maintain a YANG module called "iana-tls-cipher-suite-algs" that shadows the "TLS Cipher Suites" sub-registry of the "Transport Layer Security (TLS) Parameters" registry [IANA-CIPHER-ALGS].

This registry defines a YANG identity for each cipher suite algorithm, and a "base" identity from which all of the other identities are derived.

An initial version of this module can be found in Appendix A.1.

- \* Please note that this module was created on June 2st, 2021, and that additional entries may have been added in the interim before this document's publication. If this is that case, IANA may either publish just an updated module containing the new entries, or publish the initial module as is immediately followed by a "revision" containing the additional algorithm names.
- \* Please also note that the "status" statement has been set to "deprecated", if the "RECOMMENDED" column in the registry had the value 'N', and to "obsolete", if the "References" column included Moving single-DES and IDEA TLS ciphersuites to Historic (<https://datatracker.ietf.org/doc/status-change-tls-des-idea-ciphers-to-historic>) reference.

## 7. References

### 7.1. Normative References

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

Watsen, K., "YANG Data Types and Groupings for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-21, 14 September 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-crypto-types-21>>.

[I-D.ietf-netconf-keystore]

Watsen, K., "A YANG Data Model for a Keystore", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-23, 14 December 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-keystore-23>>.

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

Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-16, 14 December 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-trust-anchors-16>>.

[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>>.

## 7.2. Informative References

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

Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-08, 14 December 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-http-client-server-08>>.

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

Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-24, 14 December 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-netconf-client-server-24>>.

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

Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-24, 14 December 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-restconf-client-server-24>>.

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

Watsen, K., "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-ssh-client-server-26, 14 December 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-ssh-client-server-26>>.

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

Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tcp-client-server-11, 14 December 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tcp-client-server-11>>.

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

Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-26, 14 December 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tls-client-server-26>>.



[I-D.ietf-tls-external-psk-guidance]

Housley, R., Hoyland, J., Sethi, M., and C. A. Wood,  
"Guidance for External PSK Usage in TLS", Work in  
Progress, Internet-Draft, draft-ietf-tls-external-psk-  
guidance-06, 4 February 2022,  
<<https://datatracker.ietf.org/doc/html/draft-ietf-tls-external-psk-guidance-06>>.

[I-D.ietf-tls-external-psk-importer]

Benjamin, D. and C. A. Wood, "Importing External PSKs for  
TLS", Work in Progress, Internet-Draft, draft-ietf-tls-  
external-psk-importer-07, 7 March 2022,  
<<https://datatracker.ietf.org/doc/html/draft-ietf-tls-external-psk-importer-07>>.

[IANA-CIPHER-ALGS]

(IANA), I. A. N. A., "IANA "TLS Cipher Suites" Sub-  
registry of the "Transport Layer Security (TLS)  
Parameters" Registry", <<https://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml#tls-parameters-4>>.

[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>>.
- [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. YANG Modules for IANA

The module contained in this section was generated by scripts using the contents of the associated sub-registry as they existed on June 2nd, 2021.

### A.1. Initial Module for the "TLS Cipher Suites" Registry

#### A.1.1. Data Model Overview

This section provides an overview of the "iana-tls-cipher-suite-algs" module in terms of its identities and protocol-accessible nodes.

##### A.1.1.1. Identities

The following diagram lists the base "identity" statements defined in the module, of which there is just one, and illustrates that all the derived identity statements are generated from the associated IANA-maintained registry [IANA-CIPHER-ALGS].

Identities:

```
+++ cipher-suite-alg-base
+++ <identity-name from IANA registry>
```

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

##### A.1.1.2. Typedefs

The following diagram illustrates the "typedef" statements defined in the "iana-tls-cipher-suite-algs" module:

## Typedefs:

```
identityref
```

```
  +-- cipher-suite-algorithm-ref
```

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

## Comments:

- \* The typedef defined in the "iana-tls-cipher-suite-algs" module extends the "identityref" type defined in [RFC7950].

## A.1.1.3. Protocol-accessible Nodes

The following tree diagram [RFC8340] lists all the protocol-accessible nodes defined in the "iana-tls-cipher-suite-alg" module:

```
module: iana-tls-cipher-suite-algs
```

```
  +--ro supported-algorithms
```

```
    +--ro supported-algorithm*  cipher-suite-algorithm-ref
```

## Comments:

- \* Protocol-accessible nodes are those nodes that are accessible when the module is "implemented", as described in Section 5.6.5 of [RFC7950].

## A.1.2. Example Usage

The following example illustrates operational state data indicating the TLS cipher suite algorithms supported by the server:

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

```
<supported-algorithms
  xmlns="urn:ietf:params:xml:ns:yang:iana-tls-cipher-suite-algs"
  xmlns:tlscsa="urn:ietf:params:xml:ns:yang:iana-tls-cipher-suite-algs">
  <supported-algorithm>tlscsa:tls-ecdh-rsa-with-aes-256-gcm-sha384</supported-algorithm>
  <supported-algorithm>tlscsa:tls-dhe-rsa-with-aes-128-cbc-sha256</supported-algorithm>
  <supported-algorithm>tlscsa:tls-rsa-with-3des-ede-cbc-sha</supported-algorithm>
  <supported-algorithm>tlscsa:tls-ecdh-rsa-with-aes-256-gcm-sha384</supported-algorithm>
  <supported-algorithm>tlscsa:tls-dhe-psk-with-chacha20-poly1305-sha256</supported-algorithm>
  <supported-algorithm>tlscsa:tls-eccp-wd-with-aes-256-gcm-sha384</supported-algorithm>
  <supported-algorithm>tlscsa:tls-psk-with-aes-256-ccm</supported-algorithm>
  <supported-algorithm>tlscsa:tls-dhe-psk-with-camellia-256-cbc-sha384</supported-algorithm>
  <supported-algorithm>tlscsa:tls-ecdh-rsa-with-aes-256-cbc-sha384</supported-algorithm>
  <supported-algorithm>tlscsa:tls-ecdh-rsa-with-3des-ede-cbc-sha</supported-algorithm>
  <supported-algorithm>tlscsa:tls-dh-dss-with-aes-128-gcm-sha256</supported-algorithm>
</supported-algorithms>
```

#### A.1.3. YANG Module

Following are the complete contents to the initial IANA-maintained YANG module. Please note that the date "2021-06-02" reflects the day on which the extraction occurred.

```
<CODE BEGINS> file "iana-tls-cipher-suite-algs@2021-06-02.yang"

module iana-tls-cipher-suite-algs {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:iana-tls-cipher-suite-algs";
  prefix tlscsa;

  organization
    "Internet Assigned Numbers Authority (IANA)";

  contact
    "Postal: ICANN"
```

12025 Waterfront Drive, Suite 300  
Los Angeles, CA 90094-2536  
United States of America  
Tel: +1 310 301 5800  
Email: [iana@iana.org](mailto:iana@iana.org)";

description

"This module defines identities for the Cipher Suite algorithms defined in the 'TLS Cipher Suites' sub-registry of the 'Transport Layer Security (TLS) Parameters' registry maintained by IANA.

Copyright (c) 2021 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

The initial version of this YANG module is part of RFC FFFF (<https://www.rfc-editor.org/info/rfcFFFF>); see the RFC itself for full legal notices.";

```
revision 2021-06-02 {  
  description  
    "Initial version";  
  reference  
    "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";  
}
```

```
// Typedefs
```

```
typedef cipher-suite-algorithm-ref {  
  type identityref {  
    base "cipher-suite-alg-base";  
  }  
  description  
    "A reference to a TLS cipher suite algorithm identifier.";  
}
```

```
// Identities
```

```
identity cipher-suite-alg-base {  
  description
```

```
    "Base identity used to identify TLS cipher suites.";
}

identity tls-null-with-null-null {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-NULL-WITH-NULL-NULL";
  reference
    "RFC 5246:
      The Transport Layer Security (TLS) Protocol Version 1.2";
}

identity tls-rsa-with-null-md5 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-RSA-WITH-NULL-MD5";
  reference
    "RFC 5246:
      The Transport Layer Security (TLS) Protocol Version 1.2";
}

identity tls-rsa-with-null-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-RSA-WITH-NULL-SHA";
  reference
    "RFC 5246:
      The Transport Layer Security (TLS) Protocol Version 1.2";
}

identity tls-rsa-export-with-rc4-40-md5 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-RSA-EXPORT-WITH-RC4-40-MD5";
  reference
    "RFC 4346:
      The TLS Protocol Version 1.1
    RFC 6347:
      Datagram Transport Layer Security version 1.2";
}

identity tls-rsa-with-rc4-128-md5 {
  base cipher-suite-alg-base;
  status deprecated;
```

```
description
  "TLS-RSA-WITH-RC4-128-MD5";
reference
  "RFC 5246:
    The Transport Layer Security (TLS) Protocol Version 1.2
  RFC 6347:
    Datagram Transport Layer Security version 1.2";
}

identity tls-rsa-with-rc4-128-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-RSA-WITH-RC4-128-SHA";
  reference
    "RFC 5246:
      The Transport Layer Security (TLS) Protocol Version 1.2
    RFC 6347:
      Datagram Transport Layer Security version 1.2";
}

identity tls-rsa-export-with-rc2-cbc-40-md5 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-RSA-EXPORT-WITH-RC2-CBC-40-MD5";
  reference
    "RFC 4346:
      The TLS Protocol Version 1.1";
}

identity tls-rsa-with-idea-cbc-sha {
  base cipher-suite-alg-base;
  status obsolete;
  description
    "TLS-RSA-WITH-IDEA-CBC-SHA";
  reference
    "RFC 5469:
      DES and IDEA Cipher Suites for
      Transport Layer Security (TLS)
    RFC 5469:
      DES and IDEA Cipher Suites for
      Transport Layer Security (TLS)";
}

identity tls-rsa-export-with-des40-cbc-sha {
  base cipher-suite-alg-base;
  status deprecated;
```

```
    description
      "TLS-RSA-EXPORT-WITH-DES40-CBC-SHA";
    reference
      "RFC 4346:
        The TLS Protocol Version 1.1";
  }

  identity tls-rsa-with-des-cbc-sha {
    base cipher-suite-alg-base;
    status obsolete;
    description
      "TLS-RSA-WITH-DES-CBC-SHA";
    reference
      "RFC 5469:
        DES and IDEA Cipher Suites for
        Transport Layer Security (TLS)
        RFC 5469:
        DES and IDEA Cipher Suites for
        Transport Layer Security (TLS)";
  }

  identity tls-rsa-with-3des-ede-cbc-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
      "TLS-RSA-WITH-3DES-EDE-CBC-SHA";
    reference
      "RFC 5246:
        The Transport Layer Security (TLS) Protocol Version 1.2";
  }

  identity tls-dh-dss-export-with-des40-cbc-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
      "TLS-DH-DSS-EXPORT-WITH-DES40-CBC-SHA";
    reference
      "RFC 4346:
        The TLS Protocol Version 1.1";
  }

  identity tls-dh-dss-with-des-cbc-sha {
    base cipher-suite-alg-base;
    status obsolete;
    description
      "TLS-DH-DSS-WITH-DES-CBC-SHA";
    reference
      "RFC 5469:
```



```
        DES and IDEA Cipher Suites for
        Transport Layer Security (TLS)
    RFC 5469:
        DES and IDEA Cipher Suites for
        Transport Layer Security (TLS)";
}

identity tls-dh-dss-with-3des-ede-cbc-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-DH-DSS-WITH-3DES-EDE-CBC-SHA";
    reference
        "RFC 5246:
        The Transport Layer Security (TLS) Protocol Version 1.2";
}

identity tls-dh-rsa-export-with-des40-cbc-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-DH-RSA-EXPORT-WITH-DES40-CBC-SHA";
    reference
        "RFC 4346:
        The TLS Protocol Version 1.1";
}

identity tls-dh-rsa-with-des-cbc-sha {
    base cipher-suite-alg-base;
    status obsolete;
    description
        "TLS-DH-RSA-WITH-DES-CBC-SHA";
    reference
        "RFC 5469:
        DES and IDEA Cipher Suites for
        Transport Layer Security (TLS)
        RFC 5469:
        DES and IDEA Cipher Suites for
        Transport Layer Security (TLS)";
}

identity tls-dh-rsa-with-3des-ede-cbc-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-DH-RSA-WITH-3DES-EDE-CBC-SHA";
    reference
        "RFC 5246:
```

```
        The Transport Layer Security (TLS) Protocol Version 1.2";
    }

    identity tls-dhe-dss-export-with-des40-cbc-sha {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-DHE-DSS-EXPORT-WITH-DES40-CBC-SHA";
        reference
            "RFC 4346:
            The TLS Protocol Version 1.1";
    }

    identity tls-dhe-dss-with-des-cbc-sha {
        base cipher-suite-alg-base;
        status obsolete;
        description
            "TLS-DHE-DSS-WITH-DES-CBC-SHA";
        reference
            "RFC 5469:
            DES and IDEA Cipher Suites for
            Transport Layer Security (TLS)
            RFC 5469:
            DES and IDEA Cipher Suites for
            Transport Layer Security (TLS)";
    }

    identity tls-dhe-dss-with-3des-edc-cbc-sha {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-DHE-DSS-WITH-3DES-EDE-CBC-SHA";
        reference
            "RFC 5246:
            The Transport Layer Security (TLS) Protocol Version 1.2";
    }

    identity tls-dhe-rsa-export-with-des40-cbc-sha {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-DHE-RSA-EXPORT-WITH-DES40-CBC-SHA";
        reference
            "RFC 4346:
            The TLS Protocol Version 1.1";
    }

    identity tls-dhe-rsa-with-des-cbc-sha {
```

```
base cipher-suite-alg-base;
status obsolete;
description
  "TLS-DHE-RSA-WITH-DES-CBC-SHA";
reference
  "RFC 5469:
    DES and IDEA Cipher Suites for
    Transport Layer Security (TLS)
  RFC 5469:
    DES and IDEA Cipher Suites for
    Transport Layer Security (TLS)";
}

identity tls-dhe-rsa-with-3des-ede-cbc-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DHE-RSA-WITH-3DES-EDE-CBC-SHA";
  reference
    "RFC 5246:
      The Transport Layer Security (TLS) Protocol Version 1.2";
}

identity tls-dh-anon-export-with-rc4-40-md5 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DH-ANON-EXPORT-WITH-RC4-40-MD5";
  reference
    "RFC 4346:
      The TLS Protocol Version 1.1
    RFC 6347:
      Datagram Transport Layer Security version 1.2";
}

identity tls-dh-anon-with-rc4-128-md5 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DH-ANON-WITH-RC4-128-MD5";
  reference
    "RFC 5246:
      The Transport Layer Security (TLS) Protocol Version 1.2
    RFC 6347:
      Datagram Transport Layer Security version 1.2";
}

identity tls-dh-anon-export-with-des40-cbc-sha {
```

```
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-DH-ANON-EXPORT-WITH-DES40-CBC-SHA";
    reference
        "RFC 4346:
        The TLS Protocol Version 1.1";
}

identity tls-dh-anon-with-des-cbc-sha {
    base cipher-suite-alg-base;
    status obsolete;
    description
        "TLS-DH-ANON-WITH-DES-CBC-SHA";
    reference
        "RFC 5469:
        DES and IDEA Cipher Suites for
        Transport Layer Security (TLS)
        RFC 5469:
        DES and IDEA Cipher Suites for
        Transport Layer Security (TLS)";
}

identity tls-dh-anon-with-3des-edc-cbc-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-DH-ANON-WITH-3DES-EDE-CBC-SHA";
    reference
        "RFC 5246:
        The Transport Layer Security (TLS) Protocol Version 1.2";
}

identity tls-krb5-with-des-cbc-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-KRB5-WITH-DES-CBC-SHA";
    reference
        "RFC 2712:
        Addition of Kerberos Cipher Suites to
        Transport Layer Security (TLS)";
}

identity tls-krb5-with-3des-edc-cbc-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
```

```
    "TLS-KRB5-WITH-3DES-EDE-CBC-SHA";
  reference
    "RFC 2712:
      Addition of Kerberos Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-krb5-with-rc4-128-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-KRB5-WITH-RC4-128-SHA";
  reference
    "RFC 2712:
      Addition of Kerberos Cipher Suites to
      Transport Layer Security (TLS)
    RFC 6347:
      Datagram Transport Layer Security version 1.2";
}

identity tls-krb5-with-idea-cbc-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-KRB5-WITH-IDEA-CBC-SHA";
  reference
    "RFC 2712:
      Addition of Kerberos Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-krb5-with-des-cbc-md5 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-KRB5-WITH-DES-CBC-MD5";
  reference
    "RFC 2712:
      Addition of Kerberos Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-krb5-with-3des-edc-cbc-md5 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-KRB5-WITH-3DES-EDE-CBC-MD5";
  reference
```

```
"RFC 2712:
    Addition of Kerberos Cipher Suites to
    Transport Layer Security (TLS)";
}

identity tls-krb5-with-rc4-128-md5 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-KRB5-WITH-RC4-128-MD5";
    reference
        "RFC 2712:
            Addition of Kerberos Cipher Suites to
            Transport Layer Security (TLS)
            RFC 6347:
            Datagram Transport Layer Security version 1.2";
}

identity tls-krb5-with-idea-cbc-md5 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-KRB5-WITH-IDEA-CBC-MD5";
    reference
        "RFC 2712:
            Addition of Kerberos Cipher Suites to
            Transport Layer Security (TLS)";
}

identity tls-krb5-export-with-des-cbc-40-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-KRB5-EXPORT-WITH-DES-CBC-40-SHA";
    reference
        "RFC 2712:
            Addition of Kerberos Cipher Suites to
            Transport Layer Security (TLS)";
}

identity tls-krb5-export-with-rc2-cbc-40-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-KRB5-EXPORT-WITH-RC2-CBC-40-SHA";
    reference
        "RFC 2712:
            Addition of Kerberos Cipher Suites to
```

```
        Transport Layer Security (TLS)";
    }

    identity tls-krb5-export-with-rc4-40-sha {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-KRB5-EXPORT-WITH-RC4-40-SHA";
        reference
            "RFC 2712:
            Addition of Kerberos Cipher Suites to
            Transport Layer Security (TLS)
            RFC 6347:
            Datagram Transport Layer Security version 1.2";
    }

    identity tls-krb5-export-with-des-cbc-40-md5 {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-KRB5-EXPORT-WITH-DES-CBC-40-MD5";
        reference
            "RFC 2712:
            Addition of Kerberos Cipher Suites to
            Transport Layer Security (TLS)";
    }

    identity tls-krb5-export-with-rc2-cbc-40-md5 {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-KRB5-EXPORT-WITH-RC2-CBC-40-MD5";
        reference
            "RFC 2712:
            Addition of Kerberos Cipher Suites to
            Transport Layer Security (TLS)";
    }

    identity tls-krb5-export-with-rc4-40-md5 {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-KRB5-EXPORT-WITH-RC4-40-MD5";
        reference
            "RFC 2712:
            Addition of Kerberos Cipher Suites to
            Transport Layer Security (TLS)
            RFC 6347:
```

```
        Datagram Transport Layer Security version 1.2";
    }

    identity tls-psk-with-null-sha {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-PSK-WITH-NULL-SHA";
        reference
            "RFC 4785:
            Pre-Shared Key Cipher Suites with NULL Encryption for
            Transport Layer Security (TLS)";
    }

    identity tls-dhe-psk-with-null-sha {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-DHE-PSK-WITH-NULL-SHA";
        reference
            "RFC 4785:
            Pre-Shared Key Cipher Suites with NULL Encryption for
            Transport Layer Security (TLS)";
    }

    identity tls-rsa-psk-with-null-sha {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-RSA-PSK-WITH-NULL-SHA";
        reference
            "RFC 4785:
            Pre-Shared Key Cipher Suites with NULL Encryption for
            Transport Layer Security (TLS)";
    }

    identity tls-rsa-with-aes-128-cbc-sha {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-RSA-WITH-AES-128-CBC-SHA";
        reference
            "RFC 5246:
            The Transport Layer Security (TLS) Protocol Version 1.2";
    }

    identity tls-dh-dss-with-aes-128-cbc-sha {
        base cipher-suite-alg-base;
```



```
    status deprecated;
    description
        "TLS-DH-DSS-WITH-AES-128-CBC-SHA";
    reference
        "RFC 5246:
        The Transport Layer Security (TLS) Protocol Version 1.2";
}

identity tls-dh-rsa-with-aes-128-cbc-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-DH-RSA-WITH-AES-128-CBC-SHA";
    reference
        "RFC 5246:
        The Transport Layer Security (TLS) Protocol Version 1.2";
}

identity tls-dhe-dss-with-aes-128-cbc-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-DHE-DSS-WITH-AES-128-CBC-SHA";
    reference
        "RFC 5246:
        The Transport Layer Security (TLS) Protocol Version 1.2";
}

identity tls-dhe-rsa-with-aes-128-cbc-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-DHE-RSA-WITH-AES-128-CBC-SHA";
    reference
        "RFC 5246:
        The Transport Layer Security (TLS) Protocol Version 1.2";
}

identity tls-dh-anon-with-aes-128-cbc-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-DH-ANON-WITH-AES-128-CBC-SHA";
    reference
        "RFC 5246:
        The Transport Layer Security (TLS) Protocol Version 1.2";
}
```

```
identity tls-rsa-with-aes-256-cbc-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-RSA-WITH-AES-256-CBC-SHA";
  reference
    "RFC 5246:
      The Transport Layer Security (TLS) Protocol Version 1.2";
}

identity tls-dh-dss-with-aes-256-cbc-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DH-DSS-WITH-AES-256-CBC-SHA";
  reference
    "RFC 5246:
      The Transport Layer Security (TLS) Protocol Version 1.2";
}

identity tls-dh-rsa-with-aes-256-cbc-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DH-RSA-WITH-AES-256-CBC-SHA";
  reference
    "RFC 5246:
      The Transport Layer Security (TLS) Protocol Version 1.2";
}

identity tls-dhe-dss-with-aes-256-cbc-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DHE-DSS-WITH-AES-256-CBC-SHA";
  reference
    "RFC 5246:
      The Transport Layer Security (TLS) Protocol Version 1.2";
}

identity tls-dhe-rsa-with-aes-256-cbc-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DHE-RSA-WITH-AES-256-CBC-SHA";
  reference
    "RFC 5246:
      The Transport Layer Security (TLS) Protocol Version 1.2";
}
```

```
}

identity tls-dh-anon-with-aes-256-cbc-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DH-ANON-WITH-AES-256-CBC-SHA";
  reference
    "RFC 5246:
      The Transport Layer Security (TLS) Protocol Version 1.2";
}

identity tls-rsa-with-null-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-RSA-WITH-NULL-SHA256";
  reference
    "RFC 5246:
      The Transport Layer Security (TLS) Protocol Version 1.2";
}

identity tls-rsa-with-aes-128-cbc-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-RSA-WITH-AES-128-CBC-SHA256";
  reference
    "RFC 5246:
      The Transport Layer Security (TLS) Protocol Version 1.2";
}

identity tls-rsa-with-aes-256-cbc-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-RSA-WITH-AES-256-CBC-SHA256";
  reference
    "RFC 5246:
      The Transport Layer Security (TLS) Protocol Version 1.2";
}

identity tls-dh-dss-with-aes-128-cbc-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DH-DSS-WITH-AES-128-CBC-SHA256";
  reference
```

```
    "RFC 5246:
      The Transport Layer Security (TLS) Protocol Version 1.2";
  }

identity tls-dh-rsa-with-aes-128-cbc-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DH-RSA-WITH-AES-128-CBC-SHA256";
  reference
    "RFC 5246:
      The Transport Layer Security (TLS) Protocol Version 1.2";
}

identity tls-dhe-dss-with-aes-128-cbc-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DHE-DSS-WITH-AES-128-CBC-SHA256";
  reference
    "RFC 5246:
      The Transport Layer Security (TLS) Protocol Version 1.2";
}

identity tls-rsa-with-camellia-128-cbc-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-RSA-WITH-CAMELLIA-128-CBC-SHA";
  reference
    "RFC 5932:
      Camellia Cipher Suites for TLS";
}

identity tls-dh-dss-with-camellia-128-cbc-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DH-DSS-WITH-CAMELLIA-128-CBC-SHA";
  reference
    "RFC 5932:
      Camellia Cipher Suites for TLS";
}

identity tls-dh-rsa-with-camellia-128-cbc-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
```

```
        "TLS-DH-RSA-WITH-CAMELLIA-128-CBC-SHA";
    reference
        "RFC 5932:
        Camellia Cipher Suites for TLS";
}

identity tls-dhe-dss-with-camellia-128-cbc-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-DHE-DSS-WITH-CAMELLIA-128-CBC-SHA";
    reference
        "RFC 5932:
        Camellia Cipher Suites for TLS";
}

identity tls-dhe-rsa-with-camellia-128-cbc-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-DHE-RSA-WITH-CAMELLIA-128-CBC-SHA";
    reference
        "RFC 5932:
        Camellia Cipher Suites for TLS";
}

identity tls-dh-anon-with-camellia-128-cbc-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-DH-ANON-WITH-CAMELLIA-128-CBC-SHA";
    reference
        "RFC 5932:
        Camellia Cipher Suites for TLS";
}

identity tls-dhe-rsa-with-aes-128-cbc-sha256 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-DHE-RSA-WITH-AES-128-CBC-SHA256";
    reference
        "RFC 5246:
        The Transport Layer Security (TLS) Protocol Version 1.2";
}

identity tls-dh-dss-with-aes-256-cbc-sha256 {
    base cipher-suite-alg-base;
```

```
    status deprecated;
    description
        "TLS-DH-DSS-WITH-AES-256-CBC-SHA256";
    reference
        "RFC 5246:
        The Transport Layer Security (TLS) Protocol Version 1.2";
}

identity tls-dh-rsa-with-aes-256-cbc-sha256 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-DH-RSA-WITH-AES-256-CBC-SHA256";
    reference
        "RFC 5246:
        The Transport Layer Security (TLS) Protocol Version 1.2";
}

identity tls-dhe-dss-with-aes-256-cbc-sha256 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-DHE-DSS-WITH-AES-256-CBC-SHA256";
    reference
        "RFC 5246:
        The Transport Layer Security (TLS) Protocol Version 1.2";
}

identity tls-dhe-rsa-with-aes-256-cbc-sha256 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-DHE-RSA-WITH-AES-256-CBC-SHA256";
    reference
        "RFC 5246:
        The Transport Layer Security (TLS) Protocol Version 1.2";
}

identity tls-dh-anon-with-aes-128-cbc-sha256 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-DH-ANON-WITH-AES-128-CBC-SHA256";
    reference
        "RFC 5246:
        The Transport Layer Security (TLS) Protocol Version 1.2";
}
```

```
identity tls-dh-anon-with-aes-256-cbc-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DH-ANON-WITH-AES-256-CBC-SHA256";
  reference
    "RFC 5246:
      The Transport Layer Security (TLS) Protocol Version 1.2";
}

identity tls-rsa-with-camellia-256-cbc-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-RSA-WITH-CAMELLIA-256-CBC-SHA";
  reference
    "RFC 5932:
      Camellia Cipher Suites for TLS";
}

identity tls-dh-dss-with-camellia-256-cbc-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DH-DSS-WITH-CAMELLIA-256-CBC-SHA";
  reference
    "RFC 5932:
      Camellia Cipher Suites for TLS";
}

identity tls-dh-rsa-with-camellia-256-cbc-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DH-RSA-WITH-CAMELLIA-256-CBC-SHA";
  reference
    "RFC 5932:
      Camellia Cipher Suites for TLS";
}

identity tls-dhe-dss-with-camellia-256-cbc-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DHE-DSS-WITH-CAMELLIA-256-CBC-SHA";
  reference
    "RFC 5932:
      Camellia Cipher Suites for TLS";
}
```

```
}

identity tls-dhe-rsa-with-camellia-256-cbc-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DHE-RSA-WITH-CAMELLIA-256-CBC-SHA";
  reference
    "RFC 5932:
      Camellia Cipher Suites for TLS";
}

identity tls-dh-anon-with-camellia-256-cbc-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DH-ANON-WITH-CAMELLIA-256-CBC-SHA";
  reference
    "RFC 5932:
      Camellia Cipher Suites for TLS";
}

identity tls-psk-with-rc4-128-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-PSK-WITH-RC4-128-SHA";
  reference
    "RFC 4279:
      Pre-Shared Key Ciphersuites for
      Transport Layer Security (TLS)
    RFC 6347:
      Datagram Transport Layer Security version 1.2";
}

identity tls-psk-with-3des-ede-cbc-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-PSK-WITH-3DES-EDE-CBC-SHA";
  reference
    "RFC 4279:
      Pre-Shared Key Ciphersuites for
      Transport Layer Security (TLS)";
}

identity tls-psk-with-aes-128-cbc-sha {
  base cipher-suite-alg-base;
```



```
    status deprecated;
    description
        "TLS-PSK-WITH-AES-128-CBC-SHA";
    reference
        "RFC 4279:
        Pre-Shared Key Ciphersuites for
        Transport Layer Security (TLS)";
}

identity tls-psk-with-aes-256-cbc-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-PSK-WITH-AES-256-CBC-SHA";
    reference
        "RFC 4279:
        Pre-Shared Key Ciphersuites for
        Transport Layer Security (TLS)";
}

identity tls-dhe-psk-with-rc4-128-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-DHE-PSK-WITH-RC4-128-SHA";
    reference
        "RFC 4279:
        Pre-Shared Key Ciphersuites for
        Transport Layer Security (TLS)
        RFC 6347:
        Datagram Transport Layer Security version 1.2";
}

identity tls-dhe-psk-with-3des-ede-cbc-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-DHE-PSK-WITH-3DES-EDE-CBC-SHA";
    reference
        "RFC 4279:
        Pre-Shared Key Ciphersuites for
        Transport Layer Security (TLS)";
}

identity tls-dhe-psk-with-aes-128-cbc-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
```

```
    "TLS-DHE-PSK-WITH-AES-128-CBC-SHA";
  reference
    "RFC 4279:
      Pre-Shared Key Ciphersuites for
      Transport Layer Security (TLS)";
}

identity tls-dhe-psk-with-aes-256-cbc-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DHE-PSK-WITH-AES-256-CBC-SHA";
  reference
    "RFC 4279:
      Pre-Shared Key Ciphersuites for
      Transport Layer Security (TLS)";
}

identity tls-rsa-psk-with-rc4-128-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-RSA-PSK-WITH-RC4-128-SHA";
  reference
    "RFC 4279:
      Pre-Shared Key Ciphersuites for
      Transport Layer Security (TLS)
    RFC 6347:
      Datagram Transport Layer Security version 1.2";
}

identity tls-rsa-psk-with-3des-edc-cbc-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-RSA-PSK-WITH-3DES-EDE-CBC-SHA";
  reference
    "RFC 4279:
      Pre-Shared Key Ciphersuites for
      Transport Layer Security (TLS)";
}

identity tls-rsa-psk-with-aes-128-cbc-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-RSA-PSK-WITH-AES-128-CBC-SHA";
  reference
```

```
    "RFC 4279:
      Pre-Shared Key Ciphersuites for
      Transport Layer Security (TLS)";
  }

  identity tls-rsa-psk-with-aes-256-cbc-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
      "TLS-RSA-PSK-WITH-AES-256-CBC-SHA";
    reference
      "RFC 4279:
        Pre-Shared Key Ciphersuites for
        Transport Layer Security (TLS)";
  }

  identity tls-rsa-with-seed-cbc-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
      "TLS-RSA-WITH-SEED-CBC-SHA";
    reference
      "RFC 4162:
        Addition of SEED Ciphersuites to
        Transport Layer Security (TLS)";
  }

  identity tls-dh-dss-with-seed-cbc-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
      "TLS-DH-DSS-WITH-SEED-CBC-SHA";
    reference
      "RFC 4162:
        Addition of SEED Ciphersuites to
        Transport Layer Security (TLS)";
  }

  identity tls-dh-rsa-with-seed-cbc-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
      "TLS-DH-RSA-WITH-SEED-CBC-SHA";
    reference
      "RFC 4162:
        Addition of SEED Ciphersuites to
        Transport Layer Security (TLS)";
  }
```

```
identity tls-dhe-dss-with-seed-cbc-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DHE-DSS-WITH-SEED-CBC-SHA";
  reference
    "RFC 4162:
      Addition of SEED Ciphersuites to
      Transport Layer Security (TLS)";
}

identity tls-dhe-rsa-with-seed-cbc-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DHE-RSA-WITH-SEED-CBC-SHA";
  reference
    "RFC 4162:
      Addition of SEED Ciphersuites to
      Transport Layer Security (TLS)";
}

identity tls-dh-anon-with-seed-cbc-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DH-ANON-WITH-SEED-CBC-SHA";
  reference
    "RFC 4162:
      Addition of SEED Ciphersuites to
      Transport Layer Security (TLS)";
}

identity tls-rsa-with-aes-128-gcm-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-RSA-WITH-AES-128-GCM-SHA256";
  reference
    "RFC 5288:
      AES-GCM Cipher Suites for TLS";
}

identity tls-rsa-with-aes-256-gcm-sha384 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-RSA-WITH-AES-256-GCM-SHA384";
```

```
    reference
      "RFC 5288:
        AES-GCM Cipher Suites for TLS";
  }

  identity tls-dhe-rsa-with-aes-128-gcm-sha256 {
    base cipher-suite-alg-base;
    description
      "TLS-DHE-RSA-WITH-AES-128-GCM-SHA256";
    reference
      "RFC 5288:
        AES-GCM Cipher Suites for TLS";
  }

  identity tls-dhe-rsa-with-aes-256-gcm-sha384 {
    base cipher-suite-alg-base;
    description
      "TLS-DHE-RSA-WITH-AES-256-GCM-SHA384";
    reference
      "RFC 5288:
        AES-GCM Cipher Suites for TLS";
  }

  identity tls-dh-rsa-with-aes-128-gcm-sha256 {
    base cipher-suite-alg-base;
    status deprecated;
    description
      "TLS-DH-RSA-WITH-AES-128-GCM-SHA256";
    reference
      "RFC 5288:
        AES-GCM Cipher Suites for TLS";
  }

  identity tls-dh-rsa-with-aes-256-gcm-sha384 {
    base cipher-suite-alg-base;
    status deprecated;
    description
      "TLS-DH-RSA-WITH-AES-256-GCM-SHA384";
    reference
      "RFC 5288:
        AES-GCM Cipher Suites for TLS";
  }

  identity tls-dhe-dss-with-aes-128-gcm-sha256 {
    base cipher-suite-alg-base;
    status deprecated;
    description
      "TLS-DHE-DSS-WITH-AES-128-GCM-SHA256";
```

```
    reference
      "RFC 5288:
        AES-GCM Cipher Suites for TLS";
  }

  identity tls-dhe-dss-with-aes-256-gcm-sha384 {
    base cipher-suite-alg-base;
    status deprecated;
    description
      "TLS-DHE-DSS-WITH-AES-256-GCM-SHA384";
    reference
      "RFC 5288:
        AES-GCM Cipher Suites for TLS";
  }

  identity tls-dh-dss-with-aes-128-gcm-sha256 {
    base cipher-suite-alg-base;
    status deprecated;
    description
      "TLS-DH-DSS-WITH-AES-128-GCM-SHA256";
    reference
      "RFC 5288:
        AES-GCM Cipher Suites for TLS";
  }

  identity tls-dh-dss-with-aes-256-gcm-sha384 {
    base cipher-suite-alg-base;
    status deprecated;
    description
      "TLS-DH-DSS-WITH-AES-256-GCM-SHA384";
    reference
      "RFC 5288:
        AES-GCM Cipher Suites for TLS";
  }

  identity tls-dh-anon-with-aes-128-gcm-sha256 {
    base cipher-suite-alg-base;
    status deprecated;
    description
      "TLS-DH-ANON-WITH-AES-128-GCM-SHA256";
    reference
      "RFC 5288:
        AES-GCM Cipher Suites for TLS";
  }

  identity tls-dh-anon-with-aes-256-gcm-sha384 {
    base cipher-suite-alg-base;
    status deprecated;
```

```
    description
      "TLS-DH-ANON-WITH-AES-256-GCM-SHA384";
    reference
      "RFC 5288:
        AES-GCM Cipher Suites for TLS";
  }

  identity tls-psk-with-aes-128-gcm-sha256 {
    base cipher-suite-alg-base;
    status deprecated;
    description
      "TLS-PSK-WITH-AES-128-GCM-SHA256";
    reference
      "RFC 5487:
        Pre-Shared Key Cipher Suites for Transport Layer Security
        (TLS) with SHA-256/384 and AES Galois Counter Mode";
  }

  identity tls-psk-with-aes-256-gcm-sha384 {
    base cipher-suite-alg-base;
    status deprecated;
    description
      "TLS-PSK-WITH-AES-256-GCM-SHA384";
    reference
      "RFC 5487:
        Pre-Shared Key Cipher Suites for Transport Layer Security
        (TLS) with SHA-256/384 and AES Galois Counter Mode";
  }

  identity tls-dhe-psk-with-aes-128-gcm-sha256 {
    base cipher-suite-alg-base;
    description
      "TLS-DHE-PSK-WITH-AES-128-GCM-SHA256";
    reference
      "RFC 5487:
        Pre-Shared Key Cipher Suites for Transport Layer Security
        (TLS) with SHA-256/384 and AES Galois Counter Mode";
  }

  identity tls-dhe-psk-with-aes-256-gcm-sha384 {
    base cipher-suite-alg-base;
    description
      "TLS-DHE-PSK-WITH-AES-256-GCM-SHA384";
    reference
      "RFC 5487:
        Pre-Shared Key Cipher Suites for Transport Layer Security
        (TLS) with SHA-256/384 and AES Galois Counter Mode";
  }
```

```
identity tls-rsa-psk-with-aes-128-gcm-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-RSA-PSK-WITH-AES-128-GCM-SHA256";
  reference
    "RFC 5487:
      Pre-Shared Key Cipher Suites for Transport Layer Security
      (TLS) with SHA-256/384 and AES Galois Counter Mode";
}

identity tls-rsa-psk-with-aes-256-gcm-sha384 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-RSA-PSK-WITH-AES-256-GCM-SHA384";
  reference
    "RFC 5487:
      Pre-Shared Key Cipher Suites for Transport Layer Security
      (TLS) with SHA-256/384 and AES Galois Counter Mode";
}

identity tls-psk-with-aes-128-cbc-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-PSK-WITH-AES-128-CBC-SHA256";
  reference
    "RFC 5487:
      Pre-Shared Key Cipher Suites for Transport Layer Security
      (TLS) with SHA-256/384 and AES Galois Counter Mode";
}

identity tls-psk-with-aes-256-cbc-sha384 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-PSK-WITH-AES-256-CBC-SHA384";
  reference
    "RFC 5487:
      Pre-Shared Key Cipher Suites for Transport Layer Security
      (TLS) with SHA-256/384 and AES Galois Counter Mode";
}

identity tls-psk-with-null-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
```



```
    "TLS-PSK-WITH-NULL-SHA256";
  reference
    "RFC 5487:
      Pre-Shared Key Cipher Suites for Transport Layer Security
      (TLS) with SHA-256/384 and AES Galois Counter Mode";
}

identity tls-psk-with-null-sha384 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-PSK-WITH-NULL-SHA384";
  reference
    "RFC 5487:
      Pre-Shared Key Cipher Suites for Transport Layer Security
      (TLS) with SHA-256/384 and AES Galois Counter Mode";
}

identity tls-dhe-psk-with-aes-128-cbc-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DHE-PSK-WITH-AES-128-CBC-SHA256";
  reference
    "RFC 5487:
      Pre-Shared Key Cipher Suites for Transport Layer Security
      (TLS) with SHA-256/384 and AES Galois Counter Mode";
}

identity tls-dhe-psk-with-aes-256-cbc-sha384 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DHE-PSK-WITH-AES-256-CBC-SHA384";
  reference
    "RFC 5487:
      Pre-Shared Key Cipher Suites for Transport Layer Security
      (TLS) with SHA-256/384 and AES Galois Counter Mode";
}

identity tls-dhe-psk-with-null-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DHE-PSK-WITH-NULL-SHA256";
  reference
    "RFC 5487:
      Pre-Shared Key Cipher Suites for Transport Layer Security
```

```
        (TLS) with SHA-256/384 and AES Galois Counter Mode";
    }

    identity tls-dhe-psk-with-null-sha384 {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-DHE-PSK-WITH-NULL-SHA384";
        reference
            "RFC 5487:
            Pre-Shared Key Cipher Suites for Transport Layer Security
            (TLS) with SHA-256/384 and AES Galois Counter Mode";
    }

    identity tls-rsa-psk-with-aes-128-cbc-sha256 {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-RSA-PSK-WITH-AES-128-CBC-SHA256";
        reference
            "RFC 5487:
            Pre-Shared Key Cipher Suites for Transport Layer Security
            (TLS) with SHA-256/384 and AES Galois Counter Mode";
    }

    identity tls-rsa-psk-with-aes-256-cbc-sha384 {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-RSA-PSK-WITH-AES-256-CBC-SHA384";
        reference
            "RFC 5487:
            Pre-Shared Key Cipher Suites for Transport Layer Security
            (TLS) with SHA-256/384 and AES Galois Counter Mode";
    }

    identity tls-rsa-psk-with-null-sha256 {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-RSA-PSK-WITH-NULL-SHA256";
        reference
            "RFC 5487:
            Pre-Shared Key Cipher Suites for Transport Layer Security
            (TLS) with SHA-256/384 and AES Galois Counter Mode";
    }

    identity tls-rsa-psk-with-null-sha384 {
```

```
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-RSA-PSK-WITH-NULL-SHA384";
    reference
        "RFC 5487:
        Pre-Shared Key Cipher Suites for Transport Layer Security
        (TLS) with SHA-256/384 and AES Galois Counter Mode";
}

identity tls-rsa-with-camellia-128-cbc-sha256 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-RSA-WITH-CAMELLIA-128-CBC-SHA256";
    reference
        "RFC 5932:
        Camellia Cipher Suites for TLS";
}

identity tls-dh-dss-with-camellia-128-cbc-sha256 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-DH-DSS-WITH-CAMELLIA-128-CBC-SHA256";
    reference
        "RFC 5932:
        Camellia Cipher Suites for TLS";
}

identity tls-dh-rsa-with-camellia-128-cbc-sha256 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-DH-RSA-WITH-CAMELLIA-128-CBC-SHA256";
    reference
        "RFC 5932:
        Camellia Cipher Suites for TLS";
}

identity tls-dhe-dss-with-camellia-128-cbc-sha256 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-DHE-DSS-WITH-CAMELLIA-128-CBC-SHA256";
    reference
        "RFC 5932:
        Camellia Cipher Suites for TLS";
}
```

```
}

identity tls-dhe-rsa-with-camellia-128-cbc-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DHE-RSA-WITH-CAMELLIA-128-CBC-SHA256";
  reference
    "RFC 5932:
      Camellia Cipher Suites for TLS";
}

identity tls-dh-anon-with-camellia-128-cbc-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DH-ANON-WITH-CAMELLIA-128-CBC-SHA256";
  reference
    "RFC 5932:
      Camellia Cipher Suites for TLS";
}

identity tls-rsa-with-camellia-256-cbc-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-RSA-WITH-CAMELLIA-256-CBC-SHA256";
  reference
    "RFC 5932:
      Camellia Cipher Suites for TLS";
}

identity tls-dh-dss-with-camellia-256-cbc-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DH-DSS-WITH-CAMELLIA-256-CBC-SHA256";
  reference
    "RFC 5932:
      Camellia Cipher Suites for TLS";
}

identity tls-dh-rsa-with-camellia-256-cbc-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DH-RSA-WITH-CAMELLIA-256-CBC-SHA256";
  reference
```

```
        "RFC 5932:
          Camellia Cipher Suites for TLS";
    }

    identity tls-dhe-dss-with-camellia-256-cbc-sha256 {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-DHE-DSS-WITH-CAMELLIA-256-CBC-SHA256";
        reference
            "RFC 5932:
              Camellia Cipher Suites for TLS";
    }

    identity tls-dhe-rsa-with-camellia-256-cbc-sha256 {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-DHE-RSA-WITH-CAMELLIA-256-CBC-SHA256";
        reference
            "RFC 5932:
              Camellia Cipher Suites for TLS";
    }

    identity tls-dh-anon-with-camellia-256-cbc-sha256 {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-DH-ANON-WITH-CAMELLIA-256-CBC-SHA256";
        reference
            "RFC 5932:
              Camellia Cipher Suites for TLS";
    }

    identity tls-sm4-gcm-sm3 {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-SM4-GCM-SM3";
        reference
            "RFC 8998:
              ShangMi (SM) Cipher Suites for Transport Layer Security
              (TLS) Protocol Version 1.3";
    }

    identity tls-sm4-ccm-sm3 {
        base cipher-suite-alg-base;
        status deprecated;
```

```
    description
      "TLS-SM4-CCM-SM3";
    reference
      "RFC 8998:
      ShangMi (SM) Cipher Suites for Transport Layer Security
      (TLS) Protocol Version 1.3";
  }

  identity tls-empty-renegotiation-info-scsv {
    base cipher-suite-alg-base;
    status deprecated;
    description
      "TLS-EMPTY-RENEGOTIATION-INFO-SCSV";
    reference
      "RFC 5746:
      Transport Layer Security (TLS)
      Renegotiation Indication Extension";
  }

  identity tls-aes-128-gcm-sha256 {
    base cipher-suite-alg-base;
    description
      "TLS-AES-128-GCM-SHA256";
    reference
      "RFC 8446:
      The Transport Layer Security (TLS) Protocol Version 1.3";
  }

  identity tls-aes-256-gcm-sha384 {
    base cipher-suite-alg-base;
    description
      "TLS-AES-256-GCM-SHA384";
    reference
      "RFC 8446:
      The Transport Layer Security (TLS) Protocol Version 1.3";
  }

  identity tls-chacha20-poly1305-sha256 {
    base cipher-suite-alg-base;
    description
      "TLS-CHACHA20-POLY1305-SHA256";
    reference
      "RFC 8446:
      The Transport Layer Security (TLS) Protocol Version 1.3";
  }

  identity tls-aes-128-ccm-sha256 {
    base cipher-suite-alg-base;
```

```
description
  "TLS-AES-128-CCM-SHA256";
reference
  "RFC 8446:
    The Transport Layer Security (TLS) Protocol Version 1.3";
}

identity tls-aes-128-ccm-8-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-AES-128-CCM-8-SHA256";
  reference
    "RFC 8446:
      The Transport Layer Security (TLS) Protocol Version 1.3";
}

identity tls-fallback-scsv {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-FALLBACK-SCSV";
  reference
    "RFC 7507:
      TLS Fallback Signaling Cipher Suite Value (SCSV)
      for Preventing Protocol Downgrade Attacks";
}

identity tls-ecdh-ecdsa-with-null-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECDH-ECDSA-WITH-NULL-SHA";
  reference
    "RFC 8422:
      Elliptic Curve Cryptography (ECC) Cipher Suites for
      Transport Layer Security (TLS) Versions 1.2 and Earlier";
}

identity tls-ecdh-ecdsa-with-rc4-128-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECDH-ECDSA-WITH-RC4-128-SHA";
  reference
    "RFC 8422:
      Elliptic Curve Cryptography (ECC) Cipher Suites for
      Transport Layer Security (TLS) Versions 1.2 and Earlier";
}
```

```
    RFC 6347:
      Datagram Transport Layer Security version 1.2";
  }

  identity tls-ecdh-ecdsa-with-3des-ede-cbc-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
      "TLS-ECDH-ECDSA-WITH-3DES-EDE-CBC-SHA";
    reference
      "RFC 8422:
        Elliptic Curve Cryptography (ECC) Cipher Suites for
        Transport Layer Security (TLS) Versions 1.2 and Earlier";
  }

  identity tls-ecdh-ecdsa-with-aes-128-cbc-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
      "TLS-ECDH-ECDSA-WITH-AES-128-CBC-SHA";
    reference
      "RFC 8422:
        Elliptic Curve Cryptography (ECC) Cipher Suites for
        Transport Layer Security (TLS) Versions 1.2 and Earlier";
  }

  identity tls-ecdh-ecdsa-with-aes-256-cbc-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
      "TLS-ECDH-ECDSA-WITH-AES-256-CBC-SHA";
    reference
      "RFC 8422:
        Elliptic Curve Cryptography (ECC) Cipher Suites for
        Transport Layer Security (TLS) Versions 1.2 and Earlier";
  }

  identity tls-ecdhe-ecdsa-with-null-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
      "TLS-ECDHE-ECDSA-WITH-NULL-SHA";
    reference
      "RFC 8422:
        Elliptic Curve Cryptography (ECC) Cipher Suites for
        Transport Layer Security (TLS) Versions 1.2 and Earlier";
  }
```



```
identity tls-ecdhe-ecdsa-with-rc4-128-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECDHE-ECDSA-WITH-RC4-128-SHA";
  reference
    "RFC 8422:
      Elliptic Curve Cryptography (ECC) Cipher Suites for
      Transport Layer Security (TLS) Versions 1.2 and Earlier
    RFC 6347:
      Datagram Transport Layer Security version 1.2";
}

identity tls-ecdhe-ecdsa-with-3des-ede-cbc-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECDHE-ECDSA-WITH-3DES-EDE-CBC-SHA";
  reference
    "RFC 8422:
      Elliptic Curve Cryptography (ECC) Cipher Suites for
      Transport Layer Security (TLS) Versions 1.2 and Earlier";
}

identity tls-ecdhe-ecdsa-with-aes-128-cbc-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECDHE-ECDSA-WITH-AES-128-CBC-SHA";
  reference
    "RFC 8422:
      Elliptic Curve Cryptography (ECC) Cipher Suites for
      Transport Layer Security (TLS) Versions 1.2 and Earlier";
}

identity tls-ecdhe-ecdsa-with-aes-256-cbc-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECDHE-ECDSA-WITH-AES-256-CBC-SHA";
  reference
    "RFC 8422:
      Elliptic Curve Cryptography (ECC) Cipher Suites for
      Transport Layer Security (TLS) Versions 1.2 and Earlier";
}

identity tls-ecdh-rsa-with-null-sha {
  base cipher-suite-alg-base;
```

```
    status deprecated;
    description
        "TLS-ECDH-RSA-WITH-NULL-SHA";
    reference
        "RFC 8422:
        Elliptic Curve Cryptography (ECC) Cipher Suites for
        Transport Layer Security (TLS) Versions 1.2 and Earlier";
}

identity tls-ecdh-rsa-with-rc4-128-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-ECDH-RSA-WITH-RC4-128-SHA";
    reference
        "RFC 8422:
        Elliptic Curve Cryptography (ECC) Cipher Suites for
        Transport Layer Security (TLS) Versions 1.2 and Earlier
        RFC 6347:
        Datagram Transport Layer Security version 1.2";
}

identity tls-ecdh-rsa-with-3des-ede-cbc-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-ECDH-RSA-WITH-3DES-EDE-CBC-SHA";
    reference
        "RFC 8422:
        Elliptic Curve Cryptography (ECC) Cipher Suites for
        Transport Layer Security (TLS) Versions 1.2 and Earlier";
}

identity tls-ecdh-rsa-with-aes-128-cbc-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-ECDH-RSA-WITH-AES-128-CBC-SHA";
    reference
        "RFC 8422:
        Elliptic Curve Cryptography (ECC) Cipher Suites for
        Transport Layer Security (TLS) Versions 1.2 and Earlier";
}

identity tls-ecdh-rsa-with-aes-256-cbc-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
```

```
    "TLS-ECDH-RSA-WITH-AES-256-CBC-SHA";
  reference
    "RFC 8422:
      Elliptic Curve Cryptography (ECC) Cipher Suites for
      Transport Layer Security (TLS) Versions 1.2 and Earlier";
}

identity tls-ecdhe-rsa-with-null-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECDHE-RSA-WITH-NULL-SHA";
  reference
    "RFC 8422:
      Elliptic Curve Cryptography (ECC) Cipher Suites for
      Transport Layer Security (TLS) Versions 1.2 and Earlier";
}

identity tls-ecdhe-rsa-with-rc4-128-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECDHE-RSA-WITH-RC4-128-SHA";
  reference
    "RFC 8422:
      Elliptic Curve Cryptography (ECC) Cipher Suites for
      Transport Layer Security (TLS) Versions 1.2 and Earlier
    RFC 6347:
      Datagram Transport Layer Security version 1.2";
}

identity tls-ecdhe-rsa-with-3des-ede-cbc-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECDHE-RSA-WITH-3DES-EDE-CBC-SHA";
  reference
    "RFC 8422:
      Elliptic Curve Cryptography (ECC) Cipher Suites for
      Transport Layer Security (TLS) Versions 1.2 and Earlier";
}

identity tls-ecdhe-rsa-with-aes-128-cbc-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECDHE-RSA-WITH-AES-128-CBC-SHA";
  reference
```

```
"RFC 8422:
    Elliptic Curve Cryptography (ECC) Cipher Suites for
    Transport Layer Security (TLS) Versions 1.2 and Earlier";
}

identity tls-ecdh-rsa-with-aes-256-cbc-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-ECDHE-RSA-WITH-AES-256-CBC-SHA";
    reference
        "RFC 8422:
            Elliptic Curve Cryptography (ECC) Cipher Suites for
            Transport Layer Security (TLS) Versions 1.2 and Earlier";
}

identity tls-ecdh-anon-with-null-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-ECDH-ANON-WITH-NULL-SHA";
    reference
        "RFC 8422:
            Elliptic Curve Cryptography (ECC) Cipher Suites for
            Transport Layer Security (TLS) Versions 1.2 and Earlier";
}

identity tls-ecdh-anon-with-rc4-128-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-ECDH-ANON-WITH-RC4-128-SHA";
    reference
        "RFC 8422:
            Elliptic Curve Cryptography (ECC) Cipher Suites for
            Transport Layer Security (TLS) Versions 1.2 and Earlier
        RFC 6347:
            Datagram Transport Layer Security version 1.2";
}

identity tls-ecdh-anon-with-3des-ede-cbc-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-ECDH-ANON-WITH-3DES-EDE-CBC-SHA";
    reference
        "RFC 8422:
            Elliptic Curve Cryptography (ECC) Cipher Suites for
```

```
        Transport Layer Security (TLS) Versions 1.2 and Earlier";
    }

    identity tls-ecdh-anon-with-aes-128-cbc-sha {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-ECDH-ANON-WITH-AES-128-CBC-SHA";
        reference
            "RFC 8422:
            Elliptic Curve Cryptography (ECC) Cipher Suites for
            Transport Layer Security (TLS) Versions 1.2 and Earlier";
    }

    identity tls-ecdh-anon-with-aes-256-cbc-sha {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-ECDH-ANON-WITH-AES-256-CBC-SHA";
        reference
            "RFC 8422:
            Elliptic Curve Cryptography (ECC) Cipher Suites for
            Transport Layer Security (TLS) Versions 1.2 and Earlier";
    }

    identity tls-srp-sha-with-3des-edc-cbc-sha {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-SRP-SHA-WITH-3DES-EDE-CBC-SHA";
        reference
            "RFC 5054:
            Using SRP for TLS Authentication";
    }

    identity tls-srp-sha-rsa-with-3des-edc-cbc-sha {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-SRP-SHA-RSA-WITH-3DES-EDE-CBC-SHA";
        reference
            "RFC 5054:
            Using SRP for TLS Authentication";
    }

    identity tls-srp-sha-dss-with-3des-edc-cbc-sha {
        base cipher-suite-alg-base;
        status deprecated;
```

```
    description
      "TLS-SRP-SHA-DSS-WITH-3DES-EDE-CBC-SHA";
    reference
      "RFC 5054:
        Using SRP for TLS Authentication";
  }

  identity tls-srp-sha-with-aes-128-cbc-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
      "TLS-SRP-SHA-WITH-AES-128-CBC-SHA";
    reference
      "RFC 5054:
        Using SRP for TLS Authentication";
  }

  identity tls-srp-sha-rsa-with-aes-128-cbc-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
      "TLS-SRP-SHA-RSA-WITH-AES-128-CBC-SHA";
    reference
      "RFC 5054:
        Using SRP for TLS Authentication";
  }

  identity tls-srp-sha-dss-with-aes-128-cbc-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
      "TLS-SRP-SHA-DSS-WITH-AES-128-CBC-SHA";
    reference
      "RFC 5054:
        Using SRP for TLS Authentication";
  }

  identity tls-srp-sha-with-aes-256-cbc-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
      "TLS-SRP-SHA-WITH-AES-256-CBC-SHA";
    reference
      "RFC 5054:
        Using SRP for TLS Authentication";
  }

  identity tls-srp-sha-rsa-with-aes-256-cbc-sha {
```

```
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-SRP-SHA-RSA-WITH-AES-256-CBC-SHA";
    reference
        "RFC 5054:
        Using SRP for TLS Authentication";
}

identity tls-srp-sha-dss-with-aes-256-cbc-sha {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-SRP-SHA-DSS-WITH-AES-256-CBC-SHA";
    reference
        "RFC 5054:
        Using SRP for TLS Authentication";
}

identity tls-ecdh-eccsa-with-aes-128-cbc-sha256 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "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";
}

identity tls-ecdh-eccsa-with-aes-256-cbc-sha384 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "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";
}

identity tls-ecdh-eccsa-with-aes-128-cbc-sha256 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-ECDH-ECDSA-WITH-AES-128-CBC-SHA256";
    reference
        "RFC 5289:
```

```
        TLS Elliptic Curve Cipher Suites with SHA-256/384
        and AES Galois Counter Mode";
    }

    identity tls-ecdh-ecdsa-with-aes-256-cbc-sha384 {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-ECDH-ECDSA-WITH-AES-256-CBC-SHA384";
        reference
            "RFC 5289:
            TLS Elliptic Curve Cipher Suites with SHA-256/384
            and AES Galois Counter Mode";
    }

    identity tls-ecdhe-rsa-with-aes-128-cbc-sha256 {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "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";
    }

    identity tls-ecdhe-rsa-with-aes-256-cbc-sha384 {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "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";
    }

    identity tls-ecdh-rsa-with-aes-128-cbc-sha256 {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-ECDH-RSA-WITH-AES-128-CBC-SHA256";
        reference
            "RFC 5289:
            TLS Elliptic Curve Cipher Suites with SHA-256/384
            and AES Galois Counter Mode";
    }
}
```



```
identity tls-ecdh-rsa-with-aes-256-cbc-sha384 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECDH-RSA-WITH-AES-256-CBC-SHA384";
  reference
    "RFC 5289:
      TLS Elliptic Curve Cipher Suites with SHA-256/384
      and AES Galois Counter Mode";
}

identity tls-ecdhe-ecdsa-with-aes-128-gcm-sha256 {
  base cipher-suite-alg-base;
  description
    "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";
}

identity tls-ecdhe-ecdsa-with-aes-256-gcm-sha384 {
  base cipher-suite-alg-base;
  description
    "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";
}

identity tls-ecdh-ecdsa-with-aes-128-gcm-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECDH-ECDSA-WITH-AES-128-GCM-SHA256";
  reference
    "RFC 5289:
      TLS Elliptic Curve Cipher Suites with SHA-256/384
      and AES Galois Counter Mode";
}

identity tls-ecdh-ecdsa-with-aes-256-gcm-sha384 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECDH-ECDSA-WITH-AES-256-GCM-SHA384";
  reference
```



```
base cipher-suite-alg-base;
status deprecated;
description
  "TLS-ECDHE-PSK-WITH-RC4-128-SHA";
reference
  "RFC 5489:
    ECDHE_PSK Ciphersuites for Transport Layer Security (TLS)
  RFC 6347:
    Datagram Transport Layer Security version 1.2";
}

identity tls-ecdhe-psk-with-3des-edc-cbc-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECDHE-PSK-WITH-3DES-EDE-CBC-SHA";
  reference
    "RFC 5489:
      ECDHE_PSK Ciphersuites for Transport Layer Security (TLS)";
}

identity tls-ecdhe-psk-with-aes-128-cbc-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECDHE-PSK-WITH-AES-128-CBC-SHA";
  reference
    "RFC 5489:
      ECDHE_PSK Ciphersuites for Transport Layer Security (TLS)";
}

identity tls-ecdhe-psk-with-aes-256-cbc-sha {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECDHE-PSK-WITH-AES-256-CBC-SHA";
  reference
    "RFC 5489:
      ECDHE_PSK Ciphersuites for Transport Layer Security (TLS)";
}

identity tls-ecdhe-psk-with-aes-128-cbc-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECDHE-PSK-WITH-AES-128-CBC-SHA256";
  reference
    "RFC 5489:"
```

```
        ECDHE_PSK Ciphersuites for Transport Layer Security (TLS)";
    }

    identity tls-ecdhe-psk-with-aes-256-cbc-sha384 {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-ECDHE-PSK-WITH-AES-256-CBC-SHA384";
        reference
            "RFC 5489:
            ECDHE_PSK Ciphersuites for Transport Layer Security (TLS)";
    }

    identity tls-ecdhe-psk-with-null-sha {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-ECDHE-PSK-WITH-NULL-SHA";
        reference
            "RFC 5489:
            ECDHE_PSK Ciphersuites for Transport Layer Security (TLS)";
    }

    identity tls-ecdhe-psk-with-null-sha256 {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-ECDHE-PSK-WITH-NULL-SHA256";
        reference
            "RFC 5489:
            ECDHE_PSK Ciphersuites for Transport Layer Security (TLS)";
    }

    identity tls-ecdhe-psk-with-null-sha384 {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-ECDHE-PSK-WITH-NULL-SHA384";
        reference
            "RFC 5489:
            ECDHE_PSK Ciphersuites for Transport Layer Security (TLS)";
    }

    identity tls-rsa-with-aria-128-cbc-sha256 {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-RSA-WITH-ARIA-128-CBC-SHA256";
```

```
reference
  "RFC 6209:
    Addition of the ARIA Cipher Suites to
    Transport Layer Security (TLS)";
}

identity tls-rsa-with-aria-256-cbc-sha384 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-RSA-WITH-ARIA-256-CBC-SHA384";
  reference
    "RFC 6209:
      Addition of the ARIA Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-dh-dss-with-aria-128-cbc-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DH-DSS-WITH-ARIA-128-CBC-SHA256";
  reference
    "RFC 6209:
      Addition of the ARIA Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-dh-dss-with-aria-256-cbc-sha384 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DH-DSS-WITH-ARIA-256-CBC-SHA384";
  reference
    "RFC 6209:
      Addition of the ARIA Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-dh-rsa-with-aria-128-cbc-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DH-RSA-WITH-ARIA-128-CBC-SHA256";
  reference
    "RFC 6209:
      Addition of the ARIA Cipher Suites to
      Transport Layer Security (TLS)";
}
```

```
}

identity tls-dh-rsa-with-aria-256-cbc-sha384 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DH-RSA-WITH-ARIA-256-CBC-SHA384";
  reference
    "RFC 6209:
      Addition of the ARIA Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-dhe-dss-with-aria-128-cbc-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DHE-DSS-WITH-ARIA-128-CBC-SHA256";
  reference
    "RFC 6209:
      Addition of the ARIA Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-dhe-dss-with-aria-256-cbc-sha384 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DHE-DSS-WITH-ARIA-256-CBC-SHA384";
  reference
    "RFC 6209:
      Addition of the ARIA Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-dhe-rsa-with-aria-128-cbc-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DHE-RSA-WITH-ARIA-128-CBC-SHA256";
  reference
    "RFC 6209:
      Addition of the ARIA Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-dhe-rsa-with-aria-256-cbc-sha384 {
  base cipher-suite-alg-base;
```

```
    status deprecated;
    description
        "TLS-DHE-RSA-WITH-ARIA-256-CBC-SHA384";
    reference
        "RFC 6209:
        Addition of the ARIA Cipher Suites to
        Transport Layer Security (TLS)";
}

identity tls-dh-anon-with-aria-128-cbc-sha256 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-DH-ANON-WITH-ARIA-128-CBC-SHA256";
    reference
        "RFC 6209:
        Addition of the ARIA Cipher Suites to
        Transport Layer Security (TLS)";
}

identity tls-dh-anon-with-aria-256-cbc-sha384 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-DH-ANON-WITH-ARIA-256-CBC-SHA384";
    reference
        "RFC 6209:
        Addition of the ARIA Cipher Suites to
        Transport Layer Security (TLS)";
}

identity tls-ecdhe-ecdsa-with-aria-128-cbc-sha256 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-ECDHE-ECDSA-WITH-ARIA-128-CBC-SHA256";
    reference
        "RFC 6209:
        Addition of the ARIA Cipher Suites to
        Transport Layer Security (TLS)";
}

identity tls-ecdhe-ecdsa-with-aria-256-cbc-sha384 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-ECDHE-ECDSA-WITH-ARIA-256-CBC-SHA384";
    reference
```

```
"RFC 6209:
    Addition of the ARIA Cipher Suites to
    Transport Layer Security (TLS)";
}

identity tls-ecdh-ecdsa-with-aria-128-cbc-sha256 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-ECDH-ECDSA-WITH-ARIA-128-CBC-SHA256";
    reference
        "RFC 6209:
            Addition of the ARIA Cipher Suites to
            Transport Layer Security (TLS)";
}

identity tls-ecdh-ecdsa-with-aria-256-cbc-sha384 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-ECDH-ECDSA-WITH-ARIA-256-CBC-SHA384";
    reference
        "RFC 6209:
            Addition of the ARIA Cipher Suites to
            Transport Layer Security (TLS)";
}

identity tls-ecdhe-rsa-with-aria-128-cbc-sha256 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-ECDHE-RSA-WITH-ARIA-128-CBC-SHA256";
    reference
        "RFC 6209:
            Addition of the ARIA Cipher Suites to
            Transport Layer Security (TLS)";
}

identity tls-ecdhe-rsa-with-aria-256-cbc-sha384 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-ECDHE-RSA-WITH-ARIA-256-CBC-SHA384";
    reference
        "RFC 6209:
            Addition of the ARIA Cipher Suites to
            Transport Layer Security (TLS)";
}
```



```
identity tls-ecdh-rsa-with-aria-128-cbc-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECDH-RSA-WITH-ARIA-128-CBC-SHA256";
  reference
    "RFC 6209:
      Addition of the ARIA Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-ecdh-rsa-with-aria-256-cbc-sha384 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECDH-RSA-WITH-ARIA-256-CBC-SHA384";
  reference
    "RFC 6209:
      Addition of the ARIA Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-rsa-with-aria-128-gcm-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-RSA-WITH-ARIA-128-GCM-SHA256";
  reference
    "RFC 6209:
      Addition of the ARIA Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-rsa-with-aria-256-gcm-sha384 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-RSA-WITH-ARIA-256-GCM-SHA384";
  reference
    "RFC 6209:
      Addition of the ARIA Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-dhe-rsa-with-aria-128-gcm-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
```

```
    "TLS-DHE-RSA-WITH-ARIA-128-GCM-SHA256";
  reference
    "RFC 6209:
      Addition of the ARIA Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-dhe-rsa-with-aria-256-gcm-sha384 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DHE-RSA-WITH-ARIA-256-GCM-SHA384";
  reference
    "RFC 6209:
      Addition of the ARIA Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-dh-rsa-with-aria-128-gcm-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DH-RSA-WITH-ARIA-128-GCM-SHA256";
  reference
    "RFC 6209:
      Addition of the ARIA Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-dh-rsa-with-aria-256-gcm-sha384 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DH-RSA-WITH-ARIA-256-GCM-SHA384";
  reference
    "RFC 6209:
      Addition of the ARIA Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-dhe-dss-with-aria-128-gcm-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DHE-DSS-WITH-ARIA-128-GCM-SHA256";
  reference
    "RFC 6209:
      Addition of the ARIA Cipher Suites to
```

```
        Transport Layer Security (TLS)";
    }

    identity tls-dhe-dss-with-aria-256-gcm-sha384 {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-DHE-DSS-WITH-ARIA-256-GCM-SHA384";
        reference
            "RFC 6209:
            Addition of the ARIA Cipher Suites to
            Transport Layer Security (TLS)";
    }

    identity tls-dh-dss-with-aria-128-gcm-sha256 {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-DH-DSS-WITH-ARIA-128-GCM-SHA256";
        reference
            "RFC 6209:
            Addition of the ARIA Cipher Suites to
            Transport Layer Security (TLS)";
    }

    identity tls-dh-dss-with-aria-256-gcm-sha384 {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-DH-DSS-WITH-ARIA-256-GCM-SHA384";
        reference
            "RFC 6209:
            Addition of the ARIA Cipher Suites to
            Transport Layer Security (TLS)";
    }

    identity tls-dh-anon-with-aria-128-gcm-sha256 {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-DH-ANON-WITH-ARIA-128-GCM-SHA256";
        reference
            "RFC 6209:
            Addition of the ARIA Cipher Suites to
            Transport Layer Security (TLS)";
    }

    identity tls-dh-anon-with-aria-256-gcm-sha384 {
```

```
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-DH-ANON-WITH-ARIA-256-GCM-SHA384";
    reference
        "RFC 6209:
        Addition of the ARIA Cipher Suites to
        Transport Layer Security (TLS)";
}

identity tls-ecdh-eccdsa-with-aria-128-gcm-sha256 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-ECDHE-ECDSA-WITH-ARIA-128-GCM-SHA256";
    reference
        "RFC 6209:
        Addition of the ARIA Cipher Suites to
        Transport Layer Security (TLS)";
}

identity tls-ecdh-eccdsa-with-aria-256-gcm-sha384 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-ECDHE-ECDSA-WITH-ARIA-256-GCM-SHA384";
    reference
        "RFC 6209:
        Addition of the ARIA Cipher Suites to
        Transport Layer Security (TLS)";
}

identity tls-ecdh-eccdsa-with-aria-128-gcm-sha256 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-ECDH-ECDSA-WITH-ARIA-128-GCM-SHA256";
    reference
        "RFC 6209:
        Addition of the ARIA Cipher Suites to
        Transport Layer Security (TLS)";
}

identity tls-ecdh-eccdsa-with-aria-256-gcm-sha384 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-ECDH-ECDSA-WITH-ARIA-256-GCM-SHA384";
```

```
reference
  "RFC 6209:
    Addition of the ARIA Cipher Suites to
    Transport Layer Security (TLS)";
}

identity tls-ecdh-rsa-with-aria-128-gcm-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECDHE-RSA-WITH-ARIA-128-GCM-SHA256";
  reference
    "RFC 6209:
      Addition of the ARIA Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-ecdh-rsa-with-aria-256-gcm-sha384 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECDHE-RSA-WITH-ARIA-256-GCM-SHA384";
  reference
    "RFC 6209:
      Addition of the ARIA Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-ecdh-rsa-with-aria-128-gcm-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECDH-RSA-WITH-ARIA-128-GCM-SHA256";
  reference
    "RFC 6209:
      Addition of the ARIA Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-ecdh-rsa-with-aria-256-gcm-sha384 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECDH-RSA-WITH-ARIA-256-GCM-SHA384";
  reference
    "RFC 6209:
      Addition of the ARIA Cipher Suites to
      Transport Layer Security (TLS)";
}
```

```
}

identity tls-psk-with-aria-128-cbc-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-PSK-WITH-ARIA-128-CBC-SHA256";
  reference
    "RFC 6209:
      Addition of the ARIA Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-psk-with-aria-256-cbc-sha384 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-PSK-WITH-ARIA-256-CBC-SHA384";
  reference
    "RFC 6209:
      Addition of the ARIA Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-dhe-psk-with-aria-128-cbc-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DHE-PSK-WITH-ARIA-128-CBC-SHA256";
  reference
    "RFC 6209:
      Addition of the ARIA Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-dhe-psk-with-aria-256-cbc-sha384 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DHE-PSK-WITH-ARIA-256-CBC-SHA384";
  reference
    "RFC 6209:
      Addition of the ARIA Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-rsa-psk-with-aria-128-cbc-sha256 {
  base cipher-suite-alg-base;
```

```
    status deprecated;
    description
        "TLS-RSA-PSK-WITH-ARIA-128-CBC-SHA256";
    reference
        "RFC 6209:
        Addition of the ARIA Cipher Suites to
        Transport Layer Security (TLS)";
}

identity tls-rsa-psk-with-aria-256-cbc-sha384 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-RSA-PSK-WITH-ARIA-256-CBC-SHA384";
    reference
        "RFC 6209:
        Addition of the ARIA Cipher Suites to
        Transport Layer Security (TLS)";
}

identity tls-psk-with-aria-128-gcm-sha256 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-PSK-WITH-ARIA-128-GCM-SHA256";
    reference
        "RFC 6209:
        Addition of the ARIA Cipher Suites to
        Transport Layer Security (TLS)";
}

identity tls-psk-with-aria-256-gcm-sha384 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-PSK-WITH-ARIA-256-GCM-SHA384";
    reference
        "RFC 6209:
        Addition of the ARIA Cipher Suites to
        Transport Layer Security (TLS)";
}

identity tls-dhe-psk-with-aria-128-gcm-sha256 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-DHE-PSK-WITH-ARIA-128-GCM-SHA256";
    reference
```

```
"RFC 6209:
    Addition of the ARIA Cipher Suites to
    Transport Layer Security (TLS)";
}

identity tls-dhe-psk-with-aria-256-gcm-sha384 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-DHE-PSK-WITH-ARIA-256-GCM-SHA384";
    reference
        "RFC 6209:
            Addition of the ARIA Cipher Suites to
            Transport Layer Security (TLS)";
}

identity tls-rsa-psk-with-aria-128-gcm-sha256 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-RSA-PSK-WITH-ARIA-128-GCM-SHA256";
    reference
        "RFC 6209:
            Addition of the ARIA Cipher Suites to
            Transport Layer Security (TLS)";
}

identity tls-rsa-psk-with-aria-256-gcm-sha384 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-RSA-PSK-WITH-ARIA-256-GCM-SHA384";
    reference
        "RFC 6209:
            Addition of the ARIA Cipher Suites to
            Transport Layer Security (TLS)";
}

identity tls-ecdh-psk-with-aria-128-cbc-sha256 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-ECDHE-PSK-WITH-ARIA-128-CBC-SHA256";
    reference
        "RFC 6209:
            Addition of the ARIA Cipher Suites to
            Transport Layer Security (TLS)";
}
```



```
identity tls-ecdhe-psk-with-aria-256-cbc-sha384 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECDHE-PSK-WITH-ARIA-256-CBC-SHA384";
  reference
    "RFC 6209:
      Addition of the ARIA Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-ecdhe-ecdsa-with-camellia-128-cbc-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECDHE-ECDSA-WITH-CAMELLIA-128-CBC-SHA256";
  reference
    "RFC 6367:
      Addition of the Camellia Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-ecdhe-ecdsa-with-camellia-256-cbc-sha384 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECDHE-ECDSA-WITH-CAMELLIA-256-CBC-SHA384";
  reference
    "RFC 6367:
      Addition of the Camellia Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-ecdh-ecdsa-with-camellia-128-cbc-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECDH-ECDSA-WITH-CAMELLIA-128-CBC-SHA256";
  reference
    "RFC 6367:
      Addition of the Camellia Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-ecdh-ecdsa-with-camellia-256-cbc-sha384 {
  base cipher-suite-alg-base;
  status deprecated;
  description
```

```
    "TLS-ECDH-ECDSA-WITH-CAMELLIA-256-CBC-SHA384";
  reference
    "RFC 6367:
      Addition of the Camellia Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-ecdh-rsa-with-camellia-128-cbc-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECDHE-RSA-WITH-CAMELLIA-128-CBC-SHA256";
  reference
    "RFC 6367:
      Addition of the Camellia Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-ecdh-rsa-with-camellia-256-cbc-sha384 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECDHE-RSA-WITH-CAMELLIA-256-CBC-SHA384";
  reference
    "RFC 6367:
      Addition of the Camellia Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-ecdh-rsa-with-camellia-128-cbc-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECDH-RSA-WITH-CAMELLIA-128-CBC-SHA256";
  reference
    "RFC 6367:
      Addition of the Camellia Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-ecdh-rsa-with-camellia-256-cbc-sha384 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECDH-RSA-WITH-CAMELLIA-256-CBC-SHA384";
  reference
    "RFC 6367:
      Addition of the Camellia Cipher Suites to
```

```
        Transport Layer Security (TLS)";
    }

    identity tls-rsa-with-camellia-128-gcm-sha256 {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-RSA-WITH-CAMELLIA-128-GCM-SHA256";
        reference
            "RFC 6367:
            Addition of the Camellia Cipher Suites to
            Transport Layer Security (TLS)";
    }

    identity tls-rsa-with-camellia-256-gcm-sha384 {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-RSA-WITH-CAMELLIA-256-GCM-SHA384";
        reference
            "RFC 6367:
            Addition of the Camellia Cipher Suites to
            Transport Layer Security (TLS)";
    }

    identity tls-dhe-rsa-with-camellia-128-gcm-sha256 {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-DHE-RSA-WITH-CAMELLIA-128-GCM-SHA256";
        reference
            "RFC 6367:
            Addition of the Camellia Cipher Suites to
            Transport Layer Security (TLS)";
    }

    identity tls-dhe-rsa-with-camellia-256-gcm-sha384 {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-DHE-RSA-WITH-CAMELLIA-256-GCM-SHA384";
        reference
            "RFC 6367:
            Addition of the Camellia Cipher Suites to
            Transport Layer Security (TLS)";
    }

    identity tls-dh-rsa-with-camellia-128-gcm-sha256 {
```

```
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-DH-RSA-WITH-CAMELLIA-128-GCM-SHA256";
    reference
        "RFC 6367:
        Addition of the Camellia Cipher Suites to
        Transport Layer Security (TLS)";
}

identity tls-dh-rsa-with-camellia-256-gcm-sha384 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-DH-RSA-WITH-CAMELLIA-256-GCM-SHA384";
    reference
        "RFC 6367:
        Addition of the Camellia Cipher Suites to
        Transport Layer Security (TLS)";
}

identity tls-dhe-dss-with-camellia-128-gcm-sha256 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-DHE-DSS-WITH-CAMELLIA-128-GCM-SHA256";
    reference
        "RFC 6367:
        Addition of the Camellia Cipher Suites to
        Transport Layer Security (TLS)";
}

identity tls-dhe-dss-with-camellia-256-gcm-sha384 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-DHE-DSS-WITH-CAMELLIA-256-GCM-SHA384";
    reference
        "RFC 6367:
        Addition of the Camellia Cipher Suites to
        Transport Layer Security (TLS)";
}

identity tls-dh-dss-with-camellia-128-gcm-sha256 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-DH-DSS-WITH-CAMELLIA-128-GCM-SHA256";
```

```
reference
  "RFC 6367:
    Addition of the Camellia Cipher Suites to
    Transport Layer Security (TLS)";
}

identity tls-dh-dss-with-camellia-256-gcm-sha384 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DH-DSS-WITH-CAMELLIA-256-GCM-SHA384";
  reference
    "RFC 6367:
      Addition of the Camellia Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-dh-anon-with-camellia-128-gcm-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DH-ANON-WITH-CAMELLIA-128-GCM-SHA256";
  reference
    "RFC 6367:
      Addition of the Camellia Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-dh-anon-with-camellia-256-gcm-sha384 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DH-ANON-WITH-CAMELLIA-256-GCM-SHA384";
  reference
    "RFC 6367:
      Addition of the Camellia Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-ecdhc-eccdsa-with-camellia-128-gcm-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECDHE-ECDSA-WITH-CAMELLIA-128-GCM-SHA256";
  reference
    "RFC 6367:
      Addition of the Camellia Cipher Suites to
      Transport Layer Security (TLS)";
}
```

```
}

identity tls-ecdh-eccsa-with-camellia-256-gcm-sha384 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECDHE-ECDSA-WITH-CAMELLIA-256-GCM-SHA384";
  reference
    "RFC 6367:
      Addition of the Camellia Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-ecdh-eccsa-with-camellia-128-gcm-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECDH-ECDSA-WITH-CAMELLIA-128-GCM-SHA256";
  reference
    "RFC 6367:
      Addition of the Camellia Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-ecdh-eccsa-with-camellia-256-gcm-sha384 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECDH-ECDSA-WITH-CAMELLIA-256-GCM-SHA384";
  reference
    "RFC 6367:
      Addition of the Camellia Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-ecdh-eccsa-with-camellia-128-gcm-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECDHE-RSA-WITH-CAMELLIA-128-GCM-SHA256";
  reference
    "RFC 6367:
      Addition of the Camellia Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-ecdh-eccsa-with-camellia-256-gcm-sha384 {
  base cipher-suite-alg-base;
```

```
    status deprecated;
    description
      "TLS-ECDHE-RSA-WITH-CAMELLIA-256-GCM-SHA384";
    reference
      "RFC 6367:
        Addition of the Camellia Cipher Suites to
        Transport Layer Security (TLS)";
  }

  identity tls-ecdh-rsa-with-camellia-128-gcm-sha256 {
    base cipher-suite-alg-base;
    status deprecated;
    description
      "TLS-ECDH-RSA-WITH-CAMELLIA-128-GCM-SHA256";
    reference
      "RFC 6367:
        Addition of the Camellia Cipher Suites to
        Transport Layer Security (TLS)";
  }

  identity tls-ecdh-rsa-with-camellia-256-gcm-sha384 {
    base cipher-suite-alg-base;
    status deprecated;
    description
      "TLS-ECDH-RSA-WITH-CAMELLIA-256-GCM-SHA384";
    reference
      "RFC 6367:
        Addition of the Camellia Cipher Suites to
        Transport Layer Security (TLS)";
  }

  identity tls-psk-with-camellia-128-gcm-sha256 {
    base cipher-suite-alg-base;
    status deprecated;
    description
      "TLS-PSK-WITH-CAMELLIA-128-GCM-SHA256";
    reference
      "RFC 6367:
        Addition of the Camellia Cipher Suites to
        Transport Layer Security (TLS)";
  }

  identity tls-psk-with-camellia-256-gcm-sha384 {
    base cipher-suite-alg-base;
    status deprecated;
    description
      "TLS-PSK-WITH-CAMELLIA-256-GCM-SHA384";
    reference
```

```
"RFC 6367:
    Addition of the Camellia Cipher Suites to
    Transport Layer Security (TLS)";
}

identity tls-dhe-psk-with-camellia-128-gcm-sha256 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-DHE-PSK-WITH-CAMELLIA-128-GCM-SHA256";
    reference
        "RFC 6367:
            Addition of the Camellia Cipher Suites to
            Transport Layer Security (TLS)";
}

identity tls-dhe-psk-with-camellia-256-gcm-sha384 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-DHE-PSK-WITH-CAMELLIA-256-GCM-SHA384";
    reference
        "RFC 6367:
            Addition of the Camellia Cipher Suites to
            Transport Layer Security (TLS)";
}

identity tls-rsa-psk-with-camellia-128-gcm-sha256 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-RSA-PSK-WITH-CAMELLIA-128-GCM-SHA256";
    reference
        "RFC 6367:
            Addition of the Camellia Cipher Suites to
            Transport Layer Security (TLS)";
}

identity tls-rsa-psk-with-camellia-256-gcm-sha384 {
    base cipher-suite-alg-base;
    status deprecated;
    description
        "TLS-RSA-PSK-WITH-CAMELLIA-256-GCM-SHA384";
    reference
        "RFC 6367:
            Addition of the Camellia Cipher Suites to
            Transport Layer Security (TLS)";
}
```



```
identity tls-psk-with-camellia-128-cbc-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-PSK-WITH-CAMELLIA-128-CBC-SHA256";
  reference
    "RFC 6367:
      Addition of the Camellia Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-psk-with-camellia-256-cbc-sha384 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-PSK-WITH-CAMELLIA-256-CBC-SHA384";
  reference
    "RFC 6367:
      Addition of the Camellia Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-dhe-psk-with-camellia-128-cbc-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DHE-PSK-WITH-CAMELLIA-128-CBC-SHA256";
  reference
    "RFC 6367:
      Addition of the Camellia Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-dhe-psk-with-camellia-256-cbc-sha384 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DHE-PSK-WITH-CAMELLIA-256-CBC-SHA384";
  reference
    "RFC 6367:
      Addition of the Camellia Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-rsa-psk-with-camellia-128-cbc-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
```

```
    "TLS-RSA-PSK-WITH-CAMELLIA-128-CBC-SHA256";
  reference
    "RFC 6367:
      Addition of the Camellia Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-rsa-psk-with-camellia-256-cbc-sha384 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-RSA-PSK-WITH-CAMELLIA-256-CBC-SHA384";
  reference
    "RFC 6367:
      Addition of the Camellia Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-ecdh-psk-with-camellia-128-cbc-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECDHE-PSK-WITH-CAMELLIA-128-CBC-SHA256";
  reference
    "RFC 6367:
      Addition of the Camellia Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-ecdh-psk-with-camellia-256-cbc-sha384 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECDHE-PSK-WITH-CAMELLIA-256-CBC-SHA384";
  reference
    "RFC 6367:
      Addition of the Camellia Cipher Suites to
      Transport Layer Security (TLS)";
}

identity tls-rsa-with-aes-128-ccm {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-RSA-WITH-AES-128-CCM";
  reference
    "RFC 6655:
      AES-CCM Cipher Suites for TLS";
}
```

```
}

identity tls-rsa-with-aes-256-ccm {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-RSA-WITH-AES-256-CCM";
  reference
    "RFC 6655:
      AES-CCM Cipher Suites for TLS";
}

identity tls-dhe-rsa-with-aes-128-ccm {
  base cipher-suite-alg-base;
  description
    "TLS-DHE-RSA-WITH-AES-128-CCM";
  reference
    "RFC 6655:
      AES-CCM Cipher Suites for TLS";
}

identity tls-dhe-rsa-with-aes-256-ccm {
  base cipher-suite-alg-base;
  description
    "TLS-DHE-RSA-WITH-AES-256-CCM";
  reference
    "RFC 6655:
      AES-CCM Cipher Suites for TLS";
}

identity tls-rsa-with-aes-128-ccm-8 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-RSA-WITH-AES-128-CCM-8";
  reference
    "RFC 6655:
      AES-CCM Cipher Suites for TLS";
}

identity tls-rsa-with-aes-256-ccm-8 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-RSA-WITH-AES-256-CCM-8";
  reference
    "RFC 6655:
      AES-CCM Cipher Suites for TLS";
}
```

```
}

identity tls-dhe-rsa-with-aes-128-ccm-8 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DHE-RSA-WITH-AES-128-CCM-8";
  reference
    "RFC 6655:
      AES-CCM Cipher Suites for TLS";
}

identity tls-dhe-rsa-with-aes-256-ccm-8 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-DHE-RSA-WITH-AES-256-CCM-8";
  reference
    "RFC 6655:
      AES-CCM Cipher Suites for TLS";
}

identity tls-psk-with-aes-128-ccm {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-PSK-WITH-AES-128-CCM";
  reference
    "RFC 6655:
      AES-CCM Cipher Suites for TLS";
}

identity tls-psk-with-aes-256-ccm {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-PSK-WITH-AES-256-CCM";
  reference
    "RFC 6655:
      AES-CCM Cipher Suites for TLS";
}

identity tls-dhe-psk-with-aes-128-ccm {
  base cipher-suite-alg-base;
  description
    "TLS-DHE-PSK-WITH-AES-128-CCM";
  reference
    "RFC 6655:
```

```
        AES-CCM Cipher Suites for TLS";
    }

    identity tls-dhe-psk-with-aes-256-ccm {
        base cipher-suite-alg-base;
        description
            "TLS-DHE-PSK-WITH-AES-256-CCM";
        reference
            "RFC 6655:
             AES-CCM Cipher Suites for TLS";
    }

    identity tls-psk-with-aes-128-ccm-8 {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-PSK-WITH-AES-128-CCM-8";
        reference
            "RFC 6655:
             AES-CCM Cipher Suites for TLS";
    }

    identity tls-psk-with-aes-256-ccm-8 {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-PSK-WITH-AES-256-CCM-8";
        reference
            "RFC 6655:
             AES-CCM Cipher Suites for TLS";
    }

    identity tls-psk-dhe-with-aes-128-ccm-8 {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-PSK-DHE-WITH-AES-128-CCM-8";
        reference
            "RFC 6655:
             AES-CCM Cipher Suites for TLS";
    }

    identity tls-psk-dhe-with-aes-256-ccm-8 {
        base cipher-suite-alg-base;
        status deprecated;
        description
            "TLS-PSK-DHE-WITH-AES-256-CCM-8";
        reference
```

```
    "RFC 6655:
      AES-CCM Cipher Suites for TLS";
  }

  identity tls-ecdhe-ecdsa-with-aes-128-ccm {
    base cipher-suite-alg-base;
    status deprecated;
    description
      "TLS-ECDHE-ECDSA-WITH-AES-128-CCM";
    reference
      "RFC 7251:
        AES-CCM ECC Cipher Suites for TLS";
  }

  identity tls-ecdhe-ecdsa-with-aes-256-ccm {
    base cipher-suite-alg-base;
    status deprecated;
    description
      "TLS-ECDHE-ECDSA-WITH-AES-256-CCM";
    reference
      "RFC 7251:
        AES-CCM ECC Cipher Suites for TLS";
  }

  identity tls-ecdhe-ecdsa-with-aes-128-ccm-8 {
    base cipher-suite-alg-base;
    status deprecated;
    description
      "TLS-ECDHE-ECDSA-WITH-AES-128-CCM-8";
    reference
      "RFC 7251:
        AES-CCM ECC Cipher Suites for TLS";
  }

  identity tls-ecdhe-ecdsa-with-aes-256-ccm-8 {
    base cipher-suite-alg-base;
    status deprecated;
    description
      "TLS-ECDHE-ECDSA-WITH-AES-256-CCM-8";
    reference
      "RFC 7251:
        AES-CCM ECC Cipher Suites for TLS";
  }

  identity tls-eccpwd-with-aes-128-gcm-sha256 {
    base cipher-suite-alg-base;
    status deprecated;
    description
```

```
    "TLS-ECCPWD-WITH-AES-128-GCM-SHA256";
  reference
    "RFC 8492:
      Secure Password Ciphersuites for
      Transport Layer Security (TLS)";
}

identity tls-eccpwd-with-aes-256-gcm-sha384 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECCPWD-WITH-AES-256-GCM-SHA384";
  reference
    "RFC 8492:
      Secure Password Ciphersuites for
      Transport Layer Security (TLS)";
}

identity tls-eccpwd-with-aes-128-ccm-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECCPWD-WITH-AES-128-CCM-SHA256";
  reference
    "RFC 8492:
      Secure Password Ciphersuites for
      Transport Layer Security (TLS)";
}

identity tls-eccpwd-with-aes-256-ccm-sha384 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-ECCPWD-WITH-AES-256-CCM-SHA384";
  reference
    "RFC 8492:
      Secure Password Ciphersuites for
      Transport Layer Security (TLS)";
}

identity tls-ecdhe-rsa-with-chacha20-poly1305-sha256 {
  base cipher-suite-alg-base;
  description
    "TLS-ECDHE-RSA-WITH-CHACHA20-POLY1305-SHA256";
  reference
    "RFC 7905:
      ChaCha20-Poly1305 Cipher Suites for
      Transport Layer Security (TLS)";
}
```

```
}

identity tls-ecdhc-ecdsa-with-chacha20-poly1305-sha256 {
  base cipher-suite-alg-base;
  description
    "TLS-ECDHE-ECDSA-WITH-CHACHA20-POLY1305-SHA256";
  reference
    "RFC 7905:
      ChaCha20-Poly1305 Cipher Suites for
      Transport Layer Security (TLS)";
}

identity tls-dhe-rsa-with-chacha20-poly1305-sha256 {
  base cipher-suite-alg-base;
  description
    "TLS-DHE-RSA-WITH-CHACHA20-POLY1305-SHA256";
  reference
    "RFC 7905:
      ChaCha20-Poly1305 Cipher Suites for
      Transport Layer Security (TLS)";
}

identity tls-psk-with-chacha20-poly1305-sha256 {
  base cipher-suite-alg-base;
  status deprecated;
  description
    "TLS-PSK-WITH-CHACHA20-POLY1305-SHA256";
  reference
    "RFC 7905:
      ChaCha20-Poly1305 Cipher Suites for
      Transport Layer Security (TLS)";
}

identity tls-ecdhc-psk-with-chacha20-poly1305-sha256 {
  base cipher-suite-alg-base;
  description
    "TLS-ECDHE-PSK-WITH-CHACHA20-POLY1305-SHA256";
  reference
    "RFC 7905:
      ChaCha20-Poly1305 Cipher Suites for
      Transport Layer Security (TLS)";
}

identity tls-dhe-psk-with-chacha20-poly1305-sha256 {
  base cipher-suite-alg-base;
  description
    "TLS-DHE-PSK-WITH-CHACHA20-POLY1305-SHA256";
  reference
```



```
    "RFC 7905:
      ChaCha20-Poly1305 Cipher Suites for
      Transport Layer Security (TLS)";
  }

  identity tls-rsa-psk-with-chacha20-poly1305-sha256 {
    base cipher-suite-alg-base;
    status deprecated;
    description
      "TLS-RSA-PSK-WITH-CHACHA20-POLY1305-SHA256";
    reference
      "RFC 7905:
        ChaCha20-Poly1305 Cipher Suites for
        Transport Layer Security (TLS)";
  }

  identity tls-ecdh-psk-with-aes-128-gcm-sha256 {
    base cipher-suite-alg-base;
    description
      "TLS-ECDHE-PSK-WITH-AES-128-GCM-SHA256";
    reference
      "RFC 8442:
        ECDHE_PSK with AES-GCM and AES-CCM Cipher Suites";
  }

  identity tls-ecdh-psk-with-aes-256-gcm-sha384 {
    base cipher-suite-alg-base;
    description
      "TLS-ECDHE-PSK-WITH-AES-256-GCM-SHA384";
    reference
      "RFC 8442:
        ECDHE_PSK with AES-GCM and AES-CCM Cipher Suites";
  }

  identity tls-ecdh-psk-with-aes-128-ccm-8-sha256 {
    base cipher-suite-alg-base;
    status deprecated;
    description
      "TLS-ECDHE-PSK-WITH-AES-128-CCM-8-SHA256";
    reference
      "RFC 8442:
        ECDHE_PSK with AES-GCM and AES-CCM Cipher Suites";
  }

  identity tls-ecdh-psk-with-aes-128-ccm-sha256 {
    base cipher-suite-alg-base;
    description
      "TLS-ECDHE-PSK-WITH-AES-128-CCM-SHA256";
```

```
    reference
      "RFC 8442:
        ECDHE_PSK with AES-GCM and AES-CCM Cipher Suites";
  }

  // Protocol-accessible Nodes

  container supported-algorithms {
    config false;
    description
      "A container for a list of cipher suite algorithms supported
        by the server.";
    leaf-list supported-algorithm {
      type cipher-suite-algorithm-ref;
      description
        "A cipher suite algorithm supported by the server.";
    }
  }
}

<CODE ENDS>
```

## Appendix B. Change Log

This section is to be removed before publishing as an RFC.

### B.1. 00 to 01

- \* Noted that '0.0.0.0' and ':::' might have special meanings.
- \* Renamed "keychain" to "keystore".

### B.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.

### B.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'.

#### B.4. 03 to 04

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

#### B.5. 04 to 05

- \* Merged changes from co-author.

#### B.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.

#### B.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.

#### B.8. 07 to 08

- \* Added a number of compatibility matrices to Section 5 (thanks Frank!)

- \* Clarified that any configured "cipher-suite" values need to be compatible with the configured private key.

## B.9. 08 to 09

- \* Updated examples to reflect update to groupings defined in the keystore draft.
- \* Add TLS keepalives features and groupings.
- \* Prefixed top-level TLS grouping nodes with 'tls-' and support mashups.
- \* Updated copyright date, boilerplate template, affiliation, and folding algorithm.

## B.10. 09 to 10

- \* Reformatted the YANG modules.

## B.11. 10 to 11

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

## B.12. 11 to 12

- \* In server model, made 'client-authentication' a 'presence' node indicating that the server supports client authentication.
- \* In the server model, added a 'required-or-optional' choice to 'client-authentication' to better support protocols such as RESTCONF.

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

## B.13. 12 to 13

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

## B.14. 12 to 13

- \* Removed 'container' under 'client-identity' to match server model.
- \* Updated examples to reflect change grouping in keystore module.

## B.15. 13 to 14

- \* Removed the "certificate" container from "client-identity" in the ietf-tls-client module.
- \* Updated examples to reflect ietf-crypto-types change (e.g., identities --> enumerations)

## B.16. 14 to 15

- \* Updated "server-authentication" and "client-authentication" nodes from being a leaf of type "ts:certificates-ref" to a container that uses "ts:local-or-truststore-certs-grouping".

## B.17. 15 to 16

- \* Removed unnecessary if-feature statements in the -client and -server modules.
- \* Cleaned up some description statements in the -client and -server modules.
- \* Fixed a canonical ordering issue in ietf-tls-common detected by new pyang.

## B.18. 16 to 17

- \* Removed choice local-or-external by removing the 'external' case and flattening the 'local' case and adding a "client-auth-supported" feature.
- \* Removed choice required-or-optional.
- \* Updated examples to include the "\*-key-format" nodes.
- \* Augmented-in "must" expressions ensuring that locally-defined public-key-format are "ct:tls-public-key-format" (must expr for ref'ed keys are TBD).

## B.19. 17 to 18

- \* Removed the unused "external-client-auth-supported" feature.
- \* Made client-indentity optional, as there may be over-the-top auth instead.
- \* Added augment to uses of local-or-keystore-symmetric-key-grouping for a psk "id" node.
- \* Added missing presence container "psks" to ietf-tls-server's "client-authentication" container.
- \* Updated examples to reflect new "bag" addition to truststore.
- \* Removed feature-limited caseless 'case' statements to improve tree diagram rendering.
- \* Refined truststore/keystore groupings to ensure the key formats "must" be particular values.
- \* Switched to using truststore's new "public-key" bag (instead of separate "ssh-public-key" and "raw-public-key" bags).
- \* Updated client/server examples to cover ALL cases (local/ref x cert/raw-key/psk).

## B.20. 18 to 19

- \* Updated the "keepalives" containers in part to address Michal Vasko's request to align with RFC 8071, and in part to better align to RFC 6520.

- \* Removed algorithm-mapping tables from the "TLS Common Model" section
- \* Removed the 'algorithm' node from the examples.
- \* Renamed both "client-certs" and "server-certs" to "ee-certs"
- \* Added a "Note to Reviewers" note to first page.

B.21. 19 to 20

- \* Modified the 'must' expression in the "ietf-tls-client:server-authentication" node to cover the "raw-public-keys" and "psks" nodes also.
- \* Added a "must 'ca-certs or ee-certs or raw-public-keys or psks'" statement to the "ietf-tls-server:client-authentication" node.
- \* Added "mandatory true" to "choice auth-type" and a "presence" statement to its ancestor.
- \* Expanded "Data Model Overview section(s) [remove "wall" of tree diagrams].
- \* Moved the "ietf-tls-common" module section to proceed the other two module sections.
- \* Updated the Security Considerations section.

B.22. 20 to 21

- \* Updated examples to reflect new "cleartext-" prefix in the cryptotypes draft.

B.23. 21 to 22

- \* In both the "client-authentication" and "server-authentication" subtrees, replaced the "psks" node from being a P-container to a leaf of type "empty".
- \* Cleaned up examples (e.g., removed FIXMEs)
- \* Fixed issues found by the SecDir review of the "keystore" draft.
- \* Updated the "psk" sections in the "ietf-tls-client" and "ietf-tls-server" modules to more correctly reflect RFC 4279.

## B.24. 22 to 23

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

## B.25. 23 to 24

- \* Added missing reference to "FIPS PUB 180-4".
- \* Added identity "tls-1.3" and updated description statement in other identities indicating that the protocol version is obsolete and enabling the feature is NOT RECOMMENDED.
- \* Added XML-comment above examples explaining the reason for the unexpected top-most element's presence.
- \* Added missing "client-ident-raw-public-key" and "client-ident-psk" features.
- \* Aligned modules with `pyang -f` formatting.
- \* Fixed nits found by YANG Doctor reviews.
- \* Added a 'Contributors' section.

## B.26. 24 to 25

- \* Added TLS 1.3 references.
- \* Clarified support for various TLS protocol versions.
- \* Moved algorithms in ietf-tls-common (plus more) to IANA-maintained modules
- \* Added "config false" lists for algorithms supported by the server.
- \* Fixed issues found during YANG Doctor review.

## B.27. 25 to 26

- \* Replaced "base64encodedvalue==" with "BASE64VALUE=" in examples.
- \* Minor editorial nits

## B.28. 26 to 27

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



\* Created identityref-based typedef for the IANA alg identity base.

#### Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by first name): Alan Luchuk, Andy Bierman, Balazs Kovacs, Benoit Claise, Bert Wijnen, David Lamparter, Dhruv Dhody, Gary Wu, Henk Birkholz, Juergen Schoenwaelder, Ladislav Lhotka, Liang Xia, Martin Bjoerklund, Mehmet Ersue, Michal Vasko, Phil Shafer, Radek Krejci, Sean Turner, and Tom Petch.

#### Contributors

Special acknowledgement goes to Gary Wu who contributed the "ietf-tls-common" module, and Tom Petch who carefully ensured that references were set correctly throughout.

#### Author's Address

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

NETCONF  
Internet-Draft  
Intended status: Standards Track  
Expires: September 1, 2017

A. Clemm  
Huawei  
E. Voit  
A. Gonzalez Prieto  
A. Tripathy  
E. Nilsen-Nygaard  
Cisco Systems  
A. Bierman  
YumaWorks  
B. Lengyel  
Ericsson  
February 28, 2017

Subscribing to YANG datastore push updates  
draft-ietf-netconf-yang-push-05

Abstract

This document defines a subscription and push mechanism for YANG datastores. This mechanism allows subscriber applications to request updates from a YANG datastore, which are then pushed by the publisher to a receiver per a subscription policy, without requiring additional subscriber requests.

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 <http://datatracker.ietf.org/drafts/current/>.

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

This Internet-Draft will expire on September 1, 2017.

Copyright Notice

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

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

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

#### Table of Contents

1. Introduction . . . . .	3
2. Definitions and Acronyms . . . . .	5
3. Solution Overview . . . . .	6
3.1. Subscription Model . . . . .	6
3.2. Negotiation of Subscription Policies . . . . .	7
3.3. On-Change Considerations . . . . .	8
3.4. Data Encodings . . . . .	9
3.5. YANG object filters . . . . .	10
3.6. Push Data Stream and Transport Mapping . . . . .	10
3.7. Subscription management . . . . .	14
3.8. Other considerations . . . . .	15
4. A YANG data model for management of datastore push subscriptions . . . . .	19
4.1. Overview . . . . .	19
4.2. Filters . . . . .	25
4.3. Subscription configuration . . . . .	26
4.4. Notifications . . . . .	27
4.5. RPCs . . . . .	29
5. YANG module . . . . .	34
6. Security Considerations . . . . .	47
7. Acknowledgments . . . . .	47
8. References . . . . .	47
8.1. Normative References . . . . .	47
8.2. Informative References . . . . .	48

Appendix A. Technologies to be considered for future iterations	49
A.1. Proxy YANG Subscription when the Subscriber and Receiver are different . . . . .	49
A.2. OpState and Filters . . . . .	49
A.3. Splitting push updates . . . . .	50
A.4. Potential Subscription Parameters . . . . .	50
Appendix B. Issues that are currently being worked and resolved	51
Appendix C. Changes between revisions . . . . .	51
Authors' Addresses . . . . .	52

## 1. Introduction

YANG [RFC7950] was originally designed for the Netconf protocol [RFC6241] which focused on configuration data. However, YANG can be used to model both configuration and operational data. It is therefore reasonable to expect YANG datastores will increasingly be used to support applications that care about both.

For example, service assurance applications will need to be aware of any remote updates to configuration and operational objects. Rapid awareness of object changes will enable such things as validating and maintaining cross-network integrity and consistency, or monitoring state and key performance indicators of remote devices.

Traditional approaches to remote visibility have been built on polling. With polling, data is periodically explicitly retrieved by a client from a server to stay up-to-date. However, there are issues associated with polling-based management:

- o It introduces additional load on network, devices, and applications. Each polling cycle requires a separate yet arguably redundant request that results in an interrupt, requires parsing, consumes bandwidth.
- o It lacks robustness. Polling cycles may be missed, requests may be delayed or get lost, often particularly in cases when the network is under stress and hence exactly when the need for the data is the greatest.
- o Data may be difficult to calibrate and compare. Polling requests may undergo slight fluctuations, resulting in intervals of different lengths which makes data hard to compare. Likewise, pollers may have difficulty issuing requests that reach all devices at the same time, resulting in offset polling intervals which again make data hard to compare.

A more effective alternative to polling is when an application can request to be automatically updated on current relevant content of a

datastore. If such a request is accepted, interesting updates will subsequently be pushed from that datastore.

Dependence on polling-based management is typically considered an important shortcoming of applications that rely on MIBs polled using SNMP [RFC1157]. However, without a provision to support a push-based alternative, there is no reason to believe that management applications that operate on YANG datastores will be any more effective, as they would follow the same request/response pattern.

While YANG allows the definition of push notifications, such notifications generally indicate the occurrence of certain well-specified event conditions, such as the onset of an alarm condition or the occurrence of an error. A capability to subscribe to and deliver such pre-defined event notifications has been defined in [RFC5277]. In addition, configuration change notifications have been defined in [RFC6470]. These change notifications pertain only to configuration information, not to operational state, and convey the root of the subtree to which changes were applied along with the edits, but not the modified data nodes and their values. Furthermore, while delivery of updates using notifications is a viable option, some applications desire the ability to stream updates using other transports.

Accordingly, there is a need for a service that allows applications to dynamically subscribe to updates of a YANG datastore and that allows the publisher to push those updates, possibly using one of several delivery mechanisms. Additionally, support for subscriptions configured directly on the publisher are also useful when dynamic signaling is undesirable or unsupported. The requirements for such a service are documented in [RFC7923].

This document proposes a solution. The solution builds on top of the NETCONF WG's Subscribed Notifications draft [I-D:netconf-sub-notif]. At its core, the solution defined here supplants that work by introducing datastore push update mechanisms, and providing corresponding extensions to the event subscription model. The document also includes YANG data model augmentations which extend the model and RPCs defined within [I-D:netconf-sub-notif].

Key capabilities worth highlighting include:

- o Additions to event subscription mechanisms which allow clients to subscribe to datastore updates. The subscription allows clients to specify which data they are interested in, what types of updates (e.g., create, delete, modify), and to provide filter criteria that data must meet for updates to be sent. Furthermore, subscriptions can specify a policy that directs when updates are

provided. For example, a client may request to be updated periodically in certain intervals, or whenever data changes occur.

- o Format and contents of the YANG push updates themselves.
- o The ability for a publisher to push back on requested subscription parameters. Because not every publisher may support every requested update policy for every piece of data, it is necessary for a publisher to be able to indicate whether or not it is capable of supporting a requested subscription, and possibly allow to hints at subscription parameters which might have succeeded.
- o Subscription parameters which allow the specification of QoS extensions to address prioritization between independent streams of updates.

## 2. Definitions and Acronyms

Many of the terms in this document are defined in [I-D:netconf-sub-notif]. Please see that document for these definitions.

Data node: An instance of management information in a YANG datastore.

Data node update: A data item containing the current value/property of a Data node at the time the data node update was created.

Data record: A record containing a set of one or more data node instances and their associated values.

Datastore: A conceptual store of instantiated management information, with individual data items represented by data nodes which are arranged in hierarchical manner.

Datastream: A continuous stream of data records, each including a set of updates, i.e. data node instances and their associated values.

Data subtree: An instantiated data node and the data nodes that are hierarchically contained within it.

Push-update stream: A conceptual data stream of a datastore that streams the entire datastore contents continuously and perpetually.

Update: A data item containing the current value of a data node.

Update notification: An Event Notification including those data node update(s) to be pushed in order to meet the obligations of a single

Subscription. All included data node updates must reflect the state of a Datastore at a snapshot in time.

Update record: A representation of a data node update as a data record. An update record can be included as part of an update stream. It can also be logged for retrieval. In general, an update record will include the value/property of a data node. It may also include information about the type of data node update, i.e. whether the data node was modified/updated, or newly created, or deleted.

Update trigger: A mechanism, as specified by a Subscription Policy, that determines when a data node update is to be communicated. (e.g., a change trigger, invoked when the value of a data node changes or a data node is created or deleted, or a time trigger, invoked after the laps of a periodic time interval.)

YANG object filter: A filter that contains evaluation criteria which are evaluated against YANG objects of a subscription. An update is only published if the object meets the specified filter criteria.

YANG-Push: The subscription and push mechanism for YANG datastores that is specified in this document.

### 3. Solution Overview

This document specifies a solution for a push update subscription service. This solution supports the dynamic as well as configured subscriptions to information updates from YANG datastores. A subscription might target exposed operational and/or configuration YANG objects on a device. YANG objects 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 data model that is itself defined in YANG. This model augments the event subscription model defined in [I-D:netconf-sub-notif] and introduces new capabilities that allow subscribers to specify what to include in an update notification and what triggers such an update notification.

- o Enhancements to filters. Specifically the filter must at least identify at least one targeted yang data node/subtree. The filter may also define additional yang nodes/subtrees to include or exclude. The publisher must only send to the receiver those data node updates that can traverse applied filter. Filters can be specified "inline" as part of the subscription, or can be configured separately and referenced by a subscription in order to facilitate reuse of complex filters.

- o A subscription policy definition regarding the update trigger when to send new update notifications.
  - \* For periodic subscriptions, the trigger is defined by two parameters that defines the interval with which updates are to be pushed. These parameters are the period/interval of reporting duration, and an anchor time which can be used to calculate at which times updates needs to be assembled and sent.
  - \* For on-change subscriptions, the trigger occurs whenever a change in the subscribed information is detected. On-change subscriptions have more complex semantics that can be guided by additional parameters. Please refer also to Section 3.3.
    - + One parameter specifying the dampening period. This period is the interval which must pass before a successive update notification for the same Subscription is sent. Note that the dampening period applies to the set of all data nodes within a single subscription. This means that on the first change of an object, an update notification containing that object is sent either immediately or at the end of a dampening period already in effect.
    - + Another parameter allowing the restriction of the types of changes for which updates are sent (changes to object values, object creation or deletion events).
    - + A third parameter specifying whether or not a complete push-update with all the subscribed data should be sent at the beginning of a subscription. Such a push provides the receiver the current state, and establish the frame of reference for subsequent updates.
- o Anydata encoding for the contents of periodic and on-change push updates.

The subscription data model is described via augmentations to [I-D:netconf-sub-notif] later in this specification. It is conceivable that additional subscription parameters might be interesting. Augmentations to the subscription data model may be used for this.

### 3.2. Negotiation of Subscription Policies

A dynamic subscription request SHOULD be declined based on publisher's assessment that it may be unable to provide a filtered update notifications that would meet the terms of the request. But a



subscriber may quickly follow up with a new subscription request using different parameters.

Random guessing at different parameters should be discouraged. Therefore to minimize the number of subscription iterations between subscriber and publisher, dynamic subscriptions must support a simple negotiation between subscribers and publishers for subscription parameters. This negotiation is limited to either an establish or modify subscription request, followed by no-success response. The no-success message, where available SHOULD include in the returned error information parameters. The returned parameters provide information that, when followed, increase the likelihood of success for subsequent requests. However, there are no guarantee that subsequent requests for this subscriber will in fact be accepted.

Negotiable parameters which may be returned from a publisher beyond those from [I-D:netconf-sub-notif] include: hints at acceptable time intervals, size estimates for the number of objects which would be returned from a filter, and the names of targeted objects not found in the publisher's YANG tree.

### 3.3. On-Change Considerations

On-change subscriptions allow subscribers to subscribe to updates whenever changes to objects occur. As such, on-change subscriptions are of particular interest for data that changes relatively infrequently, yet that require applications to be notified with minimal delay when changes do occur.

On-change subscriptions tend to be more difficult to implement than periodic subscriptions. Specifically, on-change subscriptions may involve a notion of state to see if a change occurred between past and current state, or the ability to tap into changes as they occur in the underlying system. Accordingly, on-change subscriptions may not be supported by all implementations or for every object.

When an on-change subscription is requested for a datastream with a given subtree filter, where not all objects support on-change update triggers, only the objects supporting on-change will be provided. For more on how objects are so marked, see Section 3.8.5

Any updates for an on-change subscription will include only supported objects for which a change was detected and which met the filtering criteria. To avoid flooding receivers with repeated updates for fast-changing objects, or objects with oscillating values, an on-change subscription allows for the definition of a dampening period. Once an update for a given object is sent, no other updates for this particular subscription are sent until the end of the dampening

period. Values sent at the end of the dampening period are the current values of all changed objects which are current at the time the dampening period expires. Changed objects includes those which were deleted or newly created during that dampening period.

On-change subscriptions can be refined to let users subscribe only to certain types of changes, for example, only to object creations and deletions, but not to modifications of object values.

### 3.4. Data Encodings

Subscribed data is encoded in either XML or JSON format. A publisher MUST support XML encoding and MAY support JSON encoding.

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.

#### 3.4.1. Periodic Subscriptions

In a periodic subscription, the data included as part of an update corresponds to data that could have been simply retrieved using a get operation and is encoded in the same way. XML encoding rules for data nodes are defined in [RFC7950]. JSON encoding rules are defined in [RFC7951]. This encoding is valid JSON, but also has special encoding rules to identify module namespaces and provide consistent type processing of YANG data.

#### 3.4.2. On-Change Subscriptions

In an on-change subscription, updates need to indicate not only values of changed data nodes but also the types of changes that occurred since the last update, such as whether data nodes were newly created since the last update or whether they were merely modified, as well as which data nodes were deleted.

Encoding rules for data in on-change updates correspond to how data would be encoded in a YANG-patch operation as specified in [RFC8072]. The "YANG-patch" would in this case be applied to the earlier state reported by the preceding update, to result in the now-current state of YANG data. Of course, contrary to a YANG-patch operation, the data is sent from the publisher to the receiver and is not restricted to configuration data.

### 3.5. YANG object filters

Subscriptions can specify filters for subscribed data. The following filters are supported:

- o subtree-filter: A subtree filter specifies a subtree that the subscription refers to. When specified, updates will only concern data nodes from this subtree. Syntax and semantics correspond to that specified for [RFC6241] section 6.
- o xpath-filter: An XPath filter specifies an XPath expression applied to the data in an update, assuming XML-encoded data.

Only a single filter can be applied to a subscription at a time.

It is conceivable for implementations to support other filters. For example, an on-change filter might specify that changes in values should be sent only when the magnitude of the change since previous updates exceeds a certain threshold. It is possible to augment the subscription data model with additional filter types.

### 3.6. Push Data Stream and Transport Mapping

Pushing data based on a subscription could be considered analogous to a response to a data retrieval request, e.g., a "get" request. However, contrary to such a request, multiple responses to the same request may get sent over a longer period of time.

An applicable mechanism is that of a notification. There are however some specifics that need to be considered. Contrary to other notifications that are associated with alarms and unexpected event occurrences, update notifications are solicited, i.e. tied to a particular subscription which triggered the notification.

A push update notification contains several parameters:

- o A subscription correlator, referencing the name of the subscription on whose behalf the notification is sent.
- o Data nodes containing a representation of the datastore subtree(s) containing the updates. In all cases, the subtree(s) are filtered per access control rules to contain only data that the subscriber is authorized to see. For on-change subscriptions, the subtree may only contain the data nodes which have changed since the start of the previous dampening interval.

This document introduces two generic notifications: "push-update" and "push-change-update". Those notifications may be encapsulated on a

transport (e.g., NETCONF or HTTP) to carry data records with updates of datastore contents as specified by a subscription. It is possible also map notifications to other transports and encodings and use the same subscription model; however, the definition of such mappings is outside the scope of this document.

A push-update notification defines a complete update of the datastore per the terms of a subscription. This type of 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 will 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 resynch a previously synched on-change subscription. The push-update record contains a data snippet that contains an instantiated subtree with the subscribed contents. The content of the update notification is equivalent to the contents that would be obtained had the same data been explicitly retrieved using e.g., a Netconf "get"-operation, with the same filters applied.

The contents of the push-update notification conceptually represents the union of all data nodes in the yang modules supported by the publisher. However, in a YANG data model, it is not practical to model the precise data contained in the updates as part of the notification. To capture this data, a single parameter that can encapsulate the full set of subscribed datastore contents is used, not parameters that represent data nodes one at a time.

A push-change-update notification is the most common type of update for on-change subscriptions. The update record in this case contains a data snippet that indicates the full set of changes that data nodes have undergone since the last notification of YANG objects. In other words, this indicates which data nodes have been created, deleted, or have had changes to their values. The format of the data snippet follows YANG-patch [RFC8072], i.e. the same format that would be used with a YANG-patch operation to apply changes to a data tree, indicating the creates, deletes, and modifications of data nodes. Please note that as the update can include a mix of configuration and operational data

The following is an example of push notification. It contains an update for subscription 1011, including a subtree with root foo that contains a leaf, bar:

```
<notification
  xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2015-03-09T19:14:56Z</eventTime>
  <push-update
    xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
    <subscription-id>1011</subscription-id>
    <time-of-update>2015-03-09T19:14:56.233Z</time-of-update>
    <datastore-contents-xml>
      <foo>
        <bar>some_string</bar>
      </foo>
    </datastore-contents-xml>
  </push-update>
</notification>
```

Figure 1: Push example

The following is an example of an on-change notification. It contains an update for subscription 89, including a new value for a leaf called beta, which is a child of a top-level container called alpha:

```
<notification
  xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2015-03-09T19:14:56Z</eventTime>
  <push-change-update xmlns=
    "urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
    <subscription-id>89</subscription-id>
    <time-of-update>2015-03-09T19:14:56.233Z</time-of-update>
    <datastore-changes-xml>
      <alpha xmlns="http://example.com/sample-data/1.0" >
        <beta>1500</beta>
      </alpha>
    </datastore-changes-xml>
  </push-change-update>
</notification>
```

Figure 2: Push example for on change

The equivalent update when requesting json encoding:

```
<notification
  xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2015-03-09T19:14:56Z</eventTime>
  <push-change-update xmlns=
    "urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
    <subscription-id>89</subscription-id>
    <time-of-update>2015-03-09T19:14:56.233Z</time-of-update>
    <datastore-changes-json>
      {
        "ietf-yang-patch:yang-patch": {
          "patch-id": [
            null
          ],
          "edit": [
            {
              "edit-id": "edit1",
              "operation": "merge",
              "target": "/alpha/beta",
              "value": {
                "beta": 1500
              }
            }
          ]
        }
      }
    </datastore-changes-json>
  </push-change-update>
</notification>
```

Figure 3: Push example for on change with JSON

When the beta leaf is deleted, the publisher may send

```
<notification
  xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2015-03-09T19:14:56Z</eventTime>
  <push-change-update xmlns=
    "urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
    <subscription-id>89</subscription-id>
    <time-of-update>2015-03-09T19:14:56.233Z</time-of-update>
    <datastore-changes-xml>
      <alpha xmlns="http://example.com/sample-data/1.0" >
        <beta urn:ietf:params:xml:ns:netconf:base:1.0:
          operation="delete"/>
      </alpha>
    </datastore-changes-xml>
  </push-change-update>
</notification>
```

Figure 4: 2nd push example for on change update

### 3.7. Subscription management

A [I-D:netconf-sub-notif] subscription needs enhancement to support YANG Push subscription negotiation. Specifically, these enhancements are needed to signal to the subscriber why an attempt has failed.

A subscription can be rejected for multiple reasons, including the lack of authorization to establish a subscription, the lack of read authorization on the requested data node, or the inability of the publisher to provide a stream with the requested semantics. In such cases, no subscription is established. Instead, the subscription-result with the failure reason is returned as part of the RPC response. In addition, a set of alternative subscription parameters MAY be returned that would likely have resulted in acceptance of the subscription request, which the subscriber may try for a future subscription attempt.

It should be noted that a rejected subscription does not result in the generation of an rpc-reply with an rpc-error element, as neither the specification of YANG-push specific errors nor the specification of additional data parameters to be returned in an error case are supported as part of a YANG data model.

For instance, 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-yang-push:1.0">
    <stream>push-update</stream>
    <filter netconf:type="xpath"
      xmlns:ex="http://example.com/sample-data/1.0"
      select="/ex:foo"/>
    <period>500</period>
    <encoding>encode-xml</encoding>
  </establish-subscription>
</netconf:rpc>
```

Figure 5: Establish-Subscription example

the publisher might return:

```
<rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <subscription-result
    xmlns="http://urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
    error-insufficient-resources
  </subscription-result>
  <period>2000</period>
</rpc-reply>
```

Figure 6: Error response example

### 3.8. Other considerations

#### 3.8.1. Authorization

A receiver of subscription data may only be sent updates for which they have proper authorization. Data that is being pushed therefore needs to be subjected to a filter that applies all corresponding rules applicable at the time of a specific pushed update, silently removing any non-authorized data from subtrees.

The authorization model for data in YANG datastores is described in the Netconf Access Control Model [RFC6536]. However, some clarifications to that RFC are needed so that the desired access control behavior is applied to pushed updates.

One of these clarifications is that a subscription may only be established if the receiver has read access to every data node specifically named within the subscription filter.



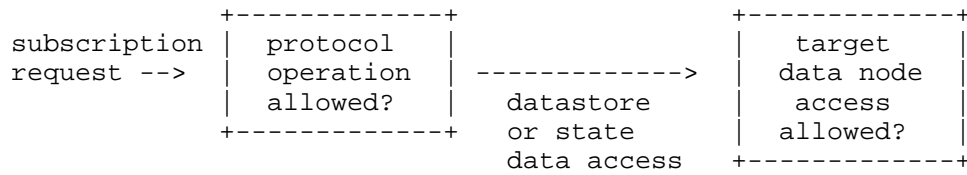


Figure 7: Access control for subscription

Likewise if a receiver no longer has read access permission to a data node named/targeted within a filter, the subscription must be abnormally terminated (with loss of access permission as the reason provided).

Another clarification to [RFC6536] is that each of the individual nodes in a pushed update must also go through access control filtering. This includes new nodes added since the last update notification, as well as existing nodes. For each of these read access must be verified. The methods of doing this efficiently are left to implementation.

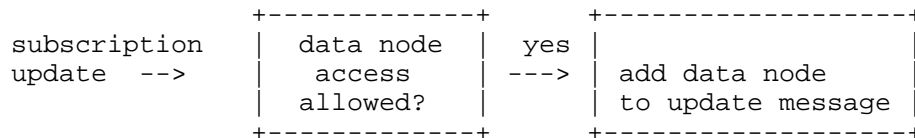


Figure 8: Access control for push updates

If there are read access control changes applied under the data node named/targeted within a filter, no notifications indicating the fact that this has occurred should be provided.

### 3.8.2. Robustness and reliability considerations

Particularly in the case of on-change push updates, it is important that push updates do not get lost.

Update notifications will typically traverse a secure and reliable transport. Notifications will not be reordered, and will also contain a time stamp. Despite these protections for on-change, it is possible that complete update notifications get lost. For this reason, update sequence numbers for push-change-updates may be included in a subscription so that an application can determine if an update has been lost.

At the same time, it is conceivable that under certain circumstances, a publisher will recognize that it is unable to include within an update notification the full set of objects desired per the terms of a subscription. In this case, the publisher must take one or more of the following actions.

- o A publisher must set the updates-not-sent flag on any update notification which is known to be missing information.
- o It may choose to suspend and resume a subscription as per [I-D:netconf-sub-notif].
- o When resuming an on-change subscription, the publisher should generate a complete patch from the previous update notification. If this is not possible and the synch-on-start option is configured, then the full datastore contents may be sent instead (effectively replacing the previous contents). If neither of these are possible, then an updates-not-sent flag must be included on the next push-change-update.

### 3.8.3. Update size and fragmentation considerations

Depending on the subscription, the volume of updates can become quite large. Additionally, based on the platform, it is possible that push-updates for a single subscription are best sent independently from different line-cards. Therefore, it may not always be practical to send the entire update in a single chunk. Implementations of push-update MAY therefore choose, at their discretion, to "chunk" updates and break them out into several push-update notifications. In this case the updates-not-sent flag will indicate that no single push-update is complete. Push-change-updates may also be chunked as long as none of the changed objects in the separate pushes are state-entangled.

### 3.8.4. Implementation considerations

Implementation specifics are outside the scope of this specification. That said, it should be noted that monitoring of operational state changes inside a system can be associated with significant implementation challenges.

Even periodic retrieval and push of operational counters may consume considerable system resources. In addition the on-change push of small amounts of configuration data may, depending on the implementation, require invocation of APIs, possibly on an object-by-object basis, possibly involving additional internal interrupts, etc.

For those reasons, it is important for an implementation to understand what subscriptions it can or cannot support. 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 in general be determined by a combination of several factors, including the subscription policy (on-change or periodic, with on-change in general being the more challenging of the two), the period in which to report changes (1 second periods will consume more resources than 1 hour periods), the amount of data in the subtree that is being subscribed to, and the number and combination of other subscriptions that are concurrently being serviced.

When providing access control to every node in a pushed update, it is possible to make and update efficient access control filters for an update. These filters can be set upon subscription and applied against a stream of updates. These filters need only be updated when (a) there is a new node added/removed from the subscribed tree with different permissions than its parent, or (b) read access permissions have been changed on nodes under the target node for the subscriber.

#### 3.8.5. Identifying on-change notifiable YANG objects

In some cases, a publisher supporting on-change notifications may not be able to push updates for some object types on-change. Reasons for this might be that the value of the data node changes frequently (e.g., a received-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 of an object type.

Support for on-change notification is usually specific to the individual YANG model and/or implementation so it is possible to define in design time. System integrators need this information (without reading any data from a live node).

The default assumption is that no data nodes support on-change notification. Schema nodes and subtrees that support on-change notifications MUST be marked as such with the YANG extension notifiable-on-change.

If the model designer wants to add the notifiable-on-change statement to an existing module, but wants to avoid modifying the text of the existing module, the notifiable-on-change statement may be added using deviation statements.

```

extension notifiable-on-change {
    Indicates whether changes to the data node are reportable in
    on-change subscriptions.

    The statement MUST only be a substatement of the leaf, leaf-list,
    container, list, anyxml, anydata statements. Zero or One
    notifiable-on-change statement is allowed per parent statement. NO
    substatements are allowed.

    The argument is a boolean value indicating whether on-change
    notifications are supported. If notifiable-on-change is not
    specified, the default is the same as the parent data node's
    value. For top level data nodes the default value is false.";

    argument value;
}

```

Figure 9: Notifiable Extension

When an on-change subscription is established data-nodes marked with notifiable-on-change false; will be automatically filtered out. This also means that authorization checks need be performed on them.

```

deviation /sys:system/sys:system-time {
    deviate add {
        yp:notifiable-on-change false;
    }
}

```

Figure 10: Deviation Example

#### 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. Following Yang tree convention in the depiction, brackets enclose list keys, "rw" means configuration, "ro" operational state data, "?" designates optional nodes, "\*" designates nodes that can have multiple instances. Parentheses with a name in the middle enclose choice and case nodes. A "+" at the end of a line indicates that the line is to be concatenated with the subsequent line. New YANG tree notation is the i] which indicates that the node in that line has been brought in / imported from another model, and an (a) which indicates this is the specific imported node augmented. In the figure below, all have been imported from 5277bis. The model consists mostly of augmentations to RPCs and notifications defined in

the data model for subscriptions for event notifications of [I-D:netconf-sub-notif].

(Note: the yp indicates augmentations from yang push above and beyond the event-notifications model)

```

module: ietf-subscribed-notifications
  +--ro streams
  |   +--ro stream*   stream
  +--rw filters
  |   +--rw filter* [identifier]
  |   |   +--rw identifier          filter-id
  |   |   +--rw (filter-type)?
  |   |   |   +---:(by-reference)
  |   |   |   |   +--rw filter-ref?          filter-ref
  |   |   |   +---:(event-filter)
  |   |   |   |   +--rw filter?
  |   |   |   +---:(yp:update-filter)
  |   |   |   |   +--rw (yp:update-filter)?
  |   |   |   |   |   +---:(yp:subtree)
  |   |   |   |   |   |   +--rw yp:subtree-filter?
  |   |   |   |   |   +---:(yp:xpath)
  |   |   |   |   |   |   +--rw yp:xpath-filter?          yang:xpath1.0
  +--rw subscription-config {configured-subscriptions}?
  |   +--rw subscription* [identifier]
  |   |   +--rw identifier          subscription-id
  |   |   +--rw stream?            stream
  |   |   +--rw encoding?          encoding
  |   |   +--rw stop-time?         yang:date-and-time
  |   |   +--rw (filter-type)?
  |   |   |   +---:(by-reference)
  |   |   |   |   +--rw filter-ref?          filter-ref
  |   |   |   +---:(event-filter)
  |   |   |   |   +--rw filter?
  |   |   |   +---:(yp:update-filter)
  |   |   |   |   +--rw (yp:update-filter)?
  |   |   |   |   |   +---:(yp:subtree)
  |   |   |   |   |   |   +--rw yp:subtree-filter?
  |   |   |   |   |   +---:(yp:xpath)
  |   |   |   |   |   |   +--rw yp:xpath-filter?          yang:xpath1.0
  +--rw receivers
  |   +--rw receiver* [address]
  |   |   +--rw address            inet:host
  |   |   +--rw port              inet:port-number
  |   |   +--rw protocol?        transport-protocol
  +--rw (notification-origin)?
  |   +---:(interface-originated)
  |   |   +--rw source-interface?    if:interface-ref
  |   +---:(address-originated)

```

```

|         |--rw source-vrf?                string
|         |--rw source-address?            inet:ip-address-no-zone
|   +--rw (yp:update-trigger)?
|       |--:(yp:periodic)
|       |   |--rw yp:period                yang:timeticks
|       |   |--rw yp:anchor-time?          yang:date-and-time
|       |--:(yp:on-change) {on-change}?
|       |   |--rw yp:dampening-period      yang:timeticks
|       |   |--rw yp:no-synch-on-start?    empty
|       |   |--rw yp:excluded-change*      change-type
|   +--rw yp:dscp?                        inet:dscp
|   +--rw yp:weighting?                   uint8
|   +--rw yp:dependency?                   sn:subscription-id
+--ro subscriptions
  +--ro subscription* [identifier]
    +--ro identifier                      subscription-id
    +--ro configured-subscription?
    |                                     empty {configured-subscriptions}?
    +--ro stream?                        stream
    +--ro encoding?                      encoding
    +--ro replay-start-time?             yang:date-and-time
    +--ro stop-time?                     yang:date-and-time
    +--ro (filter-type)?
    |   |--:(by-reference)
    |   |   +--ro filter-ref?             filter-ref
    |   |--:(event-filter)
    |   |   +--ro filter?
    |   |--:(yp:update-filter)
    |   |   +--ro (yp:update-filter)?
    |   |   |   |--:(yp:subtree)
    |   |   |   |   +--ro yp:subtree-filter?
    |   |   |   |--:(yp:xpath)
    |   |   |   |   +--ro yp:xpath-filter?          yang:xpath1.0
    +--ro (notification-origin)?
    |   |--:(interface-originated)
    |   |   +--ro source-interface?        if:interface-ref
    |   |--:(address-originated)
    |   |   +--ro source-vrf?              string
    |   |   +--ro source-address?          inet:ip-address-no-zone
    +--ro receivers
    |   +--ro receiver* [address]
    |   |   +--ro address                  inet:host
    |   |   +--ro port                    inet:port-number
    |   |   +--ro protocol?               transport-protocol
    |   |   +--ro pushed-notifications?   yang:counter64
    |   |   +--ro excluded-notifications? yang:counter64
    +--ro subscription-status?            subscription-status
    +--ro (yp:update-trigger)?

```

```

|   +---:(yp:periodic)
|   |   +---ro yp:period                yang:timeticks
|   |   +---ro yp:anchor-time?          yang:date-and-time
|   +---:(yp:on-change) {on-change}?
|   |   +---ro yp:dampening-period      yang:timeticks
|   |   +---ro yp:no-synch-on-start?    empty
|   |   +---ro yp:excluded-change*      change-type
+---ro yp:dscp?                          inet:dscp
+---ro yp:weighting?                      uint8
+---ro yp:dependency?                     sn:subscription-id

rpcs:
+---x establish-subscription
|   +---w input
|   |   +---w stream?                    stream
|   |   +---w encoding?                  encoding
|   |   +---w replay-start-time?         yang:date-and-time
|   |   +---w stop-time?                 yang:date-and-time
|   |   +---w (filter-type)?
|   |   |   +---:(by-reference)
|   |   |   |   +---w filter-ref?        filter-ref
|   |   |   +---:(event-filter)
|   |   |   |   +---w filter?
|   |   |   +---:(yp:update-filter)
|   |   |   |   +---w (yp:update-filter)?
|   |   |   |   |   +---:(yp:subtree)
|   |   |   |   |   |   +---w yp:subtree-filter?
|   |   |   |   |   +---:(yp:xpath)
|   |   |   |   |   |   +---w yp:xpath-filter?      yang:xpath1.0
|   |   +---w (yp:update-trigger)?
|   |   |   +---:(yp:periodic)
|   |   |   |   +---w yp:period                yang:timeticks
|   |   |   |   +---w yp:anchor-time?          yang:date-and-time
|   |   |   +---:(yp:on-change) {on-change}?
|   |   |   |   +---w yp:dampening-period      yang:timeticks
|   |   |   |   +---w yp:no-synch-on-start?    empty
|   |   |   |   +---w yp:excluded-change*      change-type
|   |   +---w yp:dscp?                      inet:dscp
|   |   +---w yp:weighting?                  uint8
|   |   +---w yp:dependency?                 sn:subscription-id
+---ro output
|   +---ro subscription-result            subscription-result
|   +---ro (result)?
|   |   +---:(no-success)
|   |   |   +---ro filter-failure?          string
|   |   |   +---ro replay-start-time-hint?   yang:date-and-time
|   |   |   +---ro yp:period-hint?          yang:timeticks
|   |   |   +---ro yp:error-path?           string

```

```

|         |   +--ro yp:object-count-estimate?   uint32
|         |   +--ro yp:object-count-limit?     uint32
|         |   +--ro yp:kilobytes-estimate?     uint32
|         |   +--ro yp:kilobytes-limit?       uint32
|         |   +---:(success)
|         |   +--ro identifier                  subscription-id
+---x modify-subscription
|   +---w input
|   |   +---w identifier?                      subscription-id
|   |   +---w stop-time?                      yang:date-and-time
|   |   +---w (filter-type)?
|   |   |   +---:(by-reference)
|   |   |   |   +---w filter-ref?              filter-ref
|   |   |   +---:(event-filter)
|   |   |   |   +---w filter?
|   |   |   +---:(yp:update-filter)
|   |   |   |   +---w (yp:update-filter)?
|   |   |   |   |   +---:(yp:subtree)
|   |   |   |   |   |   +---w yp:subtree-filter?
|   |   |   |   |   |   +---:(yp:xpath)
|   |   |   |   |   |   |   +---w yp:xpath-filter?      yang:xpath1.0
|   |   +---w (yp:update-trigger)?
|   |   +---:(yp:periodic)
|   |   |   +---w yp:period                    yang:timeticks
|   |   |   +---w yp:anchor-time?              yang:date-and-time
|   |   +---:(yp:on-change) {on-change}?
|   |   |   +---w yp:dampening-period          yang:timeticks
|   +---ro output
|   |   +--ro subscription-result              subscription-result
|   |   +--ro (result)?
|   |   |   +---:(no-success)
|   |   |   +--ro filter-failure?              string
|   |   |   +--ro yp:period-hint?              yang:timeticks
|   |   |   +--ro yp:error-path?              string
|   |   |   +--ro yp:object-count-estimate?   uint32
|   |   |   +--ro yp:object-count-limit?      uint32
|   |   |   +--ro yp:kilobytes-estimate?      uint32
|   |   |   +--ro yp:kilobytes-limit?         uint32
+---x delete-subscription
|   +---w input
|   |   +---w identifier                      subscription-id
|   +---ro output
|   |   +--ro subscription-result              subscription-result
+---x kill-subscription
|   +---w input
|   |   +---w identifier                      subscription-id
|   +---ro output
|   |   +--ro subscription-result              subscription-result

```



```

notifications:
  +---n replay-complete
  |   +---ro identifier      subscription-id
  +---n notification-complete
  |   +---ro identifier      subscription-id
  +---n subscription-started
  |   +---ro identifier      subscription-id
  |   +---ro stream?         stream
  |   +---ro encoding?       encoding
  |   +---ro replay-start-time? yang:date-and-time
  |   +---ro stop-time?      yang:date-and-time
  |   +---ro (filter-type)?
  |   |   +---:(by-reference)
  |   |   |   +---ro filter-ref?          filter-ref
  |   |   +---:(event-filter)
  |   |   |   +---ro filter?
  |   |   +---:(yp:update-filter)
  |   |   |   +---ro (yp:update-filter)?
  |   |   |   |   +---:(yp:subtree)
  |   |   |   |   |   +---ro yp:subtree-filter?
  |   |   |   |   |   +---:(yp:xpath)
  |   |   |   |   |   |   +---ro yp:xpath-filter?          yang:xpath1.0
  |   +---ro (yp:update-trigger)?
  |   |   +---:(yp:periodic)
  |   |   |   +---ro yp:period            yang:timeticks
  |   |   |   +---ro yp:anchor-time?      yang:date-and-time
  |   |   +---:(yp:on-change) {on-change}?
  |   |   |   +---ro yp:dampening-period  yang:timeticks
  |   |   |   +---ro yp:no-synch-on-start? empty
  |   |   |   +---ro yp:excluded-change*  change-type
  |   +---ro yp:dscp?                inet:dscp
  |   +---ro yp:weighting?            uint8
  |   +---ro yp:dependency?           sn:subscription-id
  +---n subscription-resumed
  |   +---ro identifier      subscription-id
  +---n subscription-modified
  |   +---ro identifier      subscription-id
  |   +---ro stream?         stream
  |   +---ro encoding?       encoding
  |   +---ro replay-start-time? yang:date-and-time
  |   +---ro stop-time?      yang:date-and-time
  |   +---ro (filter-type)?
  |   |   +---:(by-reference)
  |   |   |   +---ro filter-ref?          filter-ref
  |   |   +---:(event-filter)
  |   |   |   +---ro filter?
  |   |   +---:(yp:update-filter)
  |   |   |   +---ro (yp:update-filter)?

```

[illegible]

Figure 11: Model structure

Selected components of the model are summarized in the following subsections.

## 4.2. Filters

Filters can be supported via a reference to an entry in the filter container, or via direct embedding within a subscription itself. When a reference is used, it becomes possible to configure filters independently of the lifecycle of a subscription. This facilitates

the reuse of filter definitions, which can be important in case of complex filter conditions. Referenced filters can also allow an implementation to avoid evaluating filter acceptability during a dynamic subscription request.

Whether referenced or in-line, filters used for yang-push must be of case update-filters, and must follow the syntax and semantics of RFC 6241. It is not expected that implementations will support comprehensive XPATH syntax and boundless complexity. It will be up to implementations to describe what is viable, but the goal is to provide equivalent capabilities to what is available with a GET. Yang-push implementations must reject dynamic subscriptions or suspend configured subscriptions if they include filters which are unsupportable on a platform.

It is conceivable that other types of filters will be introduced for yang-push in the future. To support such filter types, additional filter cases can augment the data model.

#### 4.3. Subscription configuration

Both configured and dynamic subscriptions are represented within the list subscription-config. Each subscription has own list elements. New and enhanced parameters extending the basic subscription data model in [I-D:netconf-sub-notif] include:

- o An update filter identifying yang nodes of interest. Filter contents are specified via a reference to an existing filter, or via an in-line definition for only that subscription. The case statement differentiates the options.
- o For periodic subscriptions, triggered updates will occur at the boundaries of a specified time interval. The periodic parameters which define this interval include:
  - \* a "period" which defines duration between period push updates.
  - \* an "anchor-time". Update intervals always fall on the points in time that are a multiple of a period after the anchor time. If anchor time is not provided, then the anchor time must be set for when the initial push update can be sent.
- o When used in conjunction with period, the boundaries of periodic update periods may be calculated.
- o For on-change subscriptions, assuming the dampening period has completed, triggered 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. The first time a change is detected, 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 has passed.
  - \* an "excluded-change" flag which allows restriction of the types of changes for which updates should be sent (changes to object values, object creation or deletion events).
  - \* a "no-synch-on-start" flag which specifies whether a complete update with all the subscribed data should be sent at the beginning of a subscription.
- o Optional qos parameters to indicate the treatment of a subscription relative to other traffic between publisher and receiver. These include:
    - \* A "dscp" QoS marking which should be stamped on packets to show network QoS treatment.
    - \* A "weighting" so that bandwidth proportional to this weighting can be allocated to this subscription relative to others for that receiver.
    - \* a "dependency" upon another subscription. No push should be sent until all updates for the referenced subscription have been queued and sent.
  - o A subscription's weighting should work identically to stream dependency weighting as described within RFC 7540, section 5.3.2.
  - o A subscription's dependency should work identically to stream dependency as described within RFC 7540, sections 5.3.1, 5.3.3, and 5.3.4. If a dependency is attempted via an RPC, but the referenced subscription does not exist, the dependency will be removed.

#### 4.4. Notifications

#### 4.4.1. Monitoring and OAM Notifications

OAM notifications and mechanism are with one exception reused from [I-D:netconf-sub-notif].

The one exception is the excluded-notifications object is not applicable for yang-push. This is because discarded notifications for datastore does not have meaning in this context, and should always be zero.

#### 4.4.2. Update Notifications

The data model introduces two YANG notifications for the actual updates themselves.

Notification "push-update" is used to send a complete snapshot of the data that has been subscribed to, with all YANG object filters applied. The notification is used for periodic subscription updates in a periodic subscription.

The notification can also be used in an on-change subscription for the purposes of allowing a receiver to "synch" on a complete set of subscribed datastore contents. This will be done the start of an on-change subscription, unless no-synch-on-start is specified for that subscription. In addition, this notification MAY be used during the subscription. This might be a useful thing to do if change updates were not sent as expected (as indicated by the "updates-not-sent" flag, or an identification of loss in pushed updates), or for general resynchronization of a datastore extract at longer period intervals (such as once per day) to mitigate the possibility of any application-dependent synchronization drift. A mandatory requirement defining when to sending a push-update notification in conjunction with on-change subscription is not asserted in this specification beyond synch-on-start. However an on-change receiver must be able to handle an unsolicited push-update as a state synchronization reset.

The format and syntax of the contained update notification data corresponds to the format and syntax of data that would be returned in a corresponding get operation with the same filter parameters applied.

Notification "push-change-update" is used to send data updates for changes that have occurred in the subscribed data. This notification is used only in conjunction with on-change subscriptions.

The data updates are encoded analogous to the syntax of a corresponding yang-patch operation. It corresponds to the data that would be contained in a yang-patch operation applied to the YANG

datastore at the previous update, to result in the current state (and applying it also to operational data).

If the application detects a discontinuity in the updates it is pushing, the notification can include a flag "updates-not-sent". This is a flag which 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 in cases where it was not able to keep up with a change burst. To facilitate synchronization, a publisher MAY subsequently send a push-update containing a full snapshot of subscribed data.

#### 4.5. RPCs

YANG-Push subscriptions are established, modified, and deleted using RPCs augmented from [I-D:netconf-sub-notif].

##### 4.5.1. 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-yang-push:1.0">
    <filter netconf:type="xpath"
      xmlns:ex="http://example.com/sample-data/1.0"
      select="/ex:foo"/>
    <period>500</period>
    <encoding>encode-xml</encoding>
  </establish-subscription>
</netconf:rpc>
```

Figure 12: Establish-subscription RPC

The publisher must respond explicitly positively (i.e., subscription accepted) or negatively (i.e., subscription rejected) to the request. Positive responses include the subscription-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">
  <subscription-result
    xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
    ok
  </subscription-result>
  <subscription-id
    xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
    52
  </subscription-id>
</rpc-reply>
```

Figure 13: Establish-subscription positive RPC response

A subscription can be rejected for multiple reasons, including the lack of authorization to establish a subscription, the lack of read authorization on the requested data node, or the inability of the publisher to provide a stream with the requested semantics.

When the requester is not authorized to read the requested data node, the returned information indicates the node is unavailable. For instance, if the above request was unauthorized to read node "ex:foo" the publisher may return:

```
<rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <subscription-result
    xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
    subtree-unavailable
  </subscription-result>
  <filter-failure
    xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
    /ex:foo
  </filter-failure>
</rpc-reply>
```

Figure 14: Establish-subscription access denied response

If a request is rejected because the publisher is not able to serve it, the publisher SHOULD include in the returned error what subscription parameters would have been accepted for the request. However, there are no guarantee that subsequent requests for this subscriber or others will in fact be accepted.

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-yang-push:1.0">
    <stream>push-update</stream>
    <filter netconf:type="xpath"
      xmlns:ex="http://example.com/sample-data/1.0"
      select="/ex:foo"/>
    <dampening-period>10</dampening-period>
    <encoding>encode-xml</encoding>
  </establish-subscription>
</netconf:rpc>
```

Figure 15: 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">
  <subscription-result
    xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
    period-unsupported
  </subscription-result>
  <period-hint>100</period-hint>
</rpc-reply>
```

Figure 16: Establish-subscription error response example 2

#### 4.5.2. Modify-subscription RPC

The subscriber may send a modify-subscription RPC for a subscription previously established using RPC. The subscriber may change any subscription parameters by including the new values in the modify-subscription RPC. Parameters not included in the rpc should remain unmodified. For illustration purposes we include an exchange example where a subscriber modifies the period of the 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-yang-push:1.0">
    <stream>push-update</stream>
    <subscription-id>
      1011
    </subscription-id>
    <period>250</period>
  </modify-subscription>
</netconf:rpc>
```

Figure 17: Modify subscription request

The publisher must respond explicitly positively (i.e., subscription accepted) or negatively (i.e., subscription rejected) to the request. Positive responses include the subscription-id of the accepted subscription. In that case a publisher may respond:

```
<rpc-reply message-id="102"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <subscription-result
    xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
    ok
  </subscription-result>
</rpc-reply>
```

Figure 18: Modify subscription response

If the subscription modification is rejected, the publisher must send a response like it does for an establish-subscription and maintain the subscription as it was before the modification request. A subscription may be modified multiple times.

A configured subscription cannot be modified using modify-subscription RPC. Instead, the configuration needs to be edited as needed.

#### 4.5.3. 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-id. For example:

```
<netconf:rpc message-id="103"
  xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
  <delete-subscription
    xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
    <subscription-id>
      1011
    </subscription-id>
  </delete-subscription>
</netconf:rpc>

<rpc-reply message-id="103"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <subscription-result
    xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
    ok
  </subscription-result>
</rpc-reply>
```

Figure 19: Delete subscription

Configured subscriptions cannot be deleted via RPC, but have to be removed from the configuration.

#### 4.5.4. YANG Module Synchronization

In order to fully support datastore replication, the receiver needs to know the YANG module library that is in use by server that is being replicated. The YANG 1.0 module library information is sent by a NETCONF server in the NETCONF 'hello' message. For YANG 1.1 modules and all modules used with the RESTCONF [RFC8040] protocol, this information is provided by the YANG Library module (ietf-yang-library.yang from [RFC7895]). The YANG library information is important for the receiver to reproduce the set of object definitions used by the replicated datastore.

The YANG library includes a module list with the name, revision, enabled features, and applied deviations for each YANG module implemented by the publisher. The receiver is expected to know the YANG library information before starting a subscription. The "/modules-state/module-set-id" leaf in the "ietf-yang-library" module can be used to cache the YANG library information.

The set of modules, revisions, features, and deviations can change at run-time (if supported by the server implementation). In this case, the receiver needs to be informed of module changes before data nodes from changed modules can be processed correctly. The YANG library provides a simple "yang-library-change" notification that informs the client that the library has changed somehow. The receiver then needs

to re-read the entire YANG library data for the replicated server in order to detect the specific YANG library changes. The "ietf-netconf-notifications" module defined in [RFC6470] contains a "netconf-capability-change" notification that can identify specific module changes. For example, the module URI capability of a newly loaded module will be listed in the "added-capability" leaf-list, and the module URI capability of an removed module will be listed in the "deleted-capability" leaf-list.

## 5. YANG module

```
<CODE BEGINS> file "ietf-yang-push@2017-02-08.yang"
module ietf-yang-push {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-yang-push";
  prefix yp;

  import ietf-inet-types {
    prefix inet;
  }
  import ietf-yang-types {
    prefix yang;
  }
  import ietf-subscribed-notifications {
    prefix sn;
  }

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

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

    WG Chair: Mehmet Ersue
               <mailto:mehmet.ersue@nokia.com>

    Editor:    Alexander Clemm
               <mailto:ludwig@clemm.org>

    Editor:    Eric Voit
               <mailto:evoit@cisco.com>

    Editor:    Alberto Gonzalez Prieto
               <mailto:albertgo@cisco.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 conceptual YANG specifications
  for YANG push.";

revision 2017-02-08 {
  description
    "Updates to simplify modify-subscription, add anchor-time";
  reference
    "YANG Datastore Push, draft-ietf-netconf-yang-push-05";
}

feature on-change {
  description
    "This feature indicates that on-change updates are
    supported.";
}

/*
 * IDENTITIES
 */

/* Additional errors for subscription operations */
identity period-unsupported {
  base sn:error;
  description
    "Requested time period is too short. This can be for both
    periodic and on-change dampening.";
}

identity qos-unsupported {
  base sn:error;
  description
    "Subscription QoS parameters not supported on this platform.";
}

identity dscp-unavailable {
  base sn:error;
```

```
    description
      "Requested DSCP marking not allocatable.";
  }

  identity on-change-unsupported {
    base sn:error;
    description
      "On-change not supported.";
  }

  identity synch-on-start-unsupported {
    base sn:error;
    description
      "On-change synch-on-start not supported.";
  }

  identity synch-on-start-datatree-size {
    base sn:error;
    description
      "Synch-on-start would push a datatree which exceeds size limit.";
  }

  identity reference-mismatch {
    base sn:error;
    description
      "Mismatch in filter key and referenced yang subtree.";
  }

  identity subtree-unavailable {
    base sn:error;
    description
      "Referenced yang subtree doesn't exist, or is a node where read
      access is not permitted.";
  }

  identity datatree-size {
    base sn:error;
    description
      "Resulting push updates would exceed size limit.";
  }

  /* Additional types of streams */
  identity yang-push {
    base sn:stream;
    description
      "A conceptual datastream consisting of all datastore updates,
      including operational and configuration data.";
  }
```

```
identity custom-stream {
  base sn:stream;
  description
    "A conceptual datastream for datastore updates with custom
    updates as defined by a user.";
}

/* Additional transport option */
identity http2 {
  base sn:transport;
  description
    "HTTP2 notifications as a transport";
}

/*
 * TYPE DEFINITIONS
 */

typedef filter-id {
  type uint32;
  description
    "A type to identify filters which can be associated with a
    subscription.";
}

typedef change-type {
  type enumeration {
    enum "create" {
      description
        "A new data node was created";
    }
    enum "delete" {
      description
        "A data node was deleted";
    }
    enum "modify" {
      description
        "The value of a data node has changed";
    }
  }
  description
    "Specifies different types of changes that may occur to a
    datastore.";
}

grouping update-filter {
  description
    "This groupings defines filters for push updates for a
```

datastore tree. The filters define which updates are of interest in a push update subscription. Mixing and matching of multiple filters does not occur at the level of this grouping. When a push-update subscription is created, the filter can be a regular subscription filter, or one of the additional filters that are defined in this grouping.";

```

choice update-filter {
  description
    "Define filters regarding which data nodes to include
    in push updates";
  case subtree {
    description
      "Subtree filter.";
    anyxml subtree-filter {
      description
        "Subtree-filter used to specify the data nodes targeted
        for subscription within a subtree, or subtrees, of a
        conceptual YANG datastore. Objects matching the filter
        criteria will traverse the filter. The syntax follows
        the subtree filter syntax specified in RFC 6241.";
      reference "RFC 6241 section 6";
    }
  }
  case xpath {
    description
      "XPath filter";
    leaf xpath-filter {
      type yang:xpath1.0;
      description
        "XPath defining the data items of interest.";
    }
  }
}
}

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 {
    description
      "Defines necessary conditions for sending an event to
      the subscriber.";
    case periodic {
      description
        "The agent is requested to notify periodically the current
        values of the datastore as defined by the filter.";
    }
  }
}

```

```
    leaf period {
      type yang:timeticks;
      mandatory true;
      description
        "Duration of time which should occur between periodic
        push updates. Where the anchor of a start-time is
        available, the push will include the objects and their
        values which exist at an exact multiple of timeticks
        aligning to this start-time anchor.";
    }
    leaf anchor-time {
      type yang:date-and-time;
      description
        "Designates a timestamp from which the series of periodic
        push updates are computed. The next update will take place
        at the next period interval from the anchor time. For
        example, for an anchor time at the top of a minute and a
        period interval of a minute, the next update will be sent
        at the top of the next minute.";
    }
  }
}
case on-change {
  if-feature "on-change";
  description
    "The agent is requested to notify changes in values in the
    datastore subset as defined by a filter.";
  leaf dampening-period {
    type yang:timeticks;
    mandatory true;
    description
      "Minimum amount of time that needs to have passed since the
      last time an update was provided for the subscription.";
  }
}
}
}

grouping update-policy {
  description
    "This grouping describes the datastore specific subscription
    conditions of a subscription.";
  uses update-policy-modifiable {
    augment "update-trigger/on-change" {
      description
        "Includes objects not modifiable once subscription is
        established.";
      leaf no-synch-on-start {
        type empty;
      }
    }
  }
}
```



```

    description
        "This leaf acts as a flag that determines behavior at the
        start of the subscription. When present, synchronization
        of state at the beginning of the subscription is outside
        the scope of the subscription. Only updates about changes
        that are observed from the start time, i.e. only push-
        change-update notifications are sent. When absent (default
        behavior), in order to facilitate a receiver's
        synchronization, a full update is sent when the
        subscription starts using a push-update notification, just
        like in the case of a periodic subscription. After that,
        push-change-update notifications only are sent unless the
        Publisher chooses to resynch the subscription again.";
    }
    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 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 {
        type inet:dscp;
        default "0";
        description
            "The push update's IP packet transport priority. This is made
            visible across network hops to receiver. The transport
            priority is shared for all receivers of a given
            subscription.";
    }
    leaf weighting {
        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 {
        type sn:subscription-id;
        description
            "Provides the Subscription ID of a parent subscription which has
            absolute priority 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 of the
            parent has something ready to push.";
        reference
            "RFC-7540, section 5.3.1";
    }
}

grouping update-error-hints {
    description
        "Allow return additional negotiation hints that apply
        specifically to push updates.";
    leaf period-hint {
        type yang:timeticks;
        description
            "Returned when the requested time period is too short. This hint
            can assert an viable period for both periodic push cadence and
            on-change dampening.";
    }
    leaf error-path {
        type string;
        description
            "Reference to a YANG path which is associated with the error
            being returned.";
    }
    leaf object-count-estimate {
        type uint32;
        description
            "If there are too many objects which could potentially be
            returned by the 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
            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 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.";
  }
}

augment "/sn:establish-subscription/sn:input" {
  description
    "Define additional subscription parameters that apply
     specifically to push updates";
  uses update-policy;
  uses update-qos;
}

augment "/sn:establish-subscription/sn:input/" +
  "sn:filter-type" {
  description
    "Add push filters to selection of filter types.";
  case update-filter {
    description
      "Additional filter options for push subscription.";
    uses update-filter;
  }
}

augment "/sn:establish-subscription/sn:output/" +
  "sn:result/sn:no-success" {
  description
    "Add push datastore error info and hints to RPC output.";
  uses update-error-hints;
}

augment "/sn:modify-subscription/sn:input" {
  description
    "Define additional subscription parameters that apply
     specifically to push updates.";
  uses update-policy-modifiable;
}

augment "/sn:modify-subscription/sn:input/" +
  "sn:filter-type" {
  description
    "Add push filters to selection of filter types.";
  case update-filter {
```

```
        description
            "Additional filter options for push subscription.";
        uses update-filter;
    }
}
augment "/sn:modify-subscription/sn:output/" +
    "sn:result/sn:no-success" {
    description
        "Add push datastore error info and hints to RPC output.";
    uses update-error-hints;
}

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.";
    leaf subscription-id {
        type sn:subscription-id;
        mandatory true;
        description
            "This references the subscription because of which the
            notification is sent.";
    }
    leaf time-of-update {
        type yang:date-and-time;
        description
            "This leaf contains the time of the update.";
    }
    leaf updates-not-sent {
        type empty;
        description
            "This is a flag which indicates that not all data nodes
            subscribed to are included with this update. In other words,
            the publisher has failed to fulfill its full subscription
            obligations. This may lead to intermittent loss of
            synchronization of data at the client. Synchronization at the
            client can occur when the next push-update is received.";
    }
    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 format and syntax of the data
```

```
        corresponds to the format and syntax of data that would be
        returned in a corresponding get operation with the same
        filter parameters applied.";
    }
}
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 subscription-id {
        type sn:subscription-id;
        mandatory true;
        description
            "This references the subscription because of which the
            notification is sent.";
    }
    leaf time-of-update {
        type yang:date-and-time;
        description
            "This leaf contains the time of the update, i.e. the time at
            which the change was observed.";
    }
    leaf updates-not-sent {
        type empty;
        description
            "This is a flag which indicates that 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.
            To facilitate synchronization, a publisher MAY subsequently
            send a push-update containing a full snapshot of subscribed
            data. Such a push-update might also be triggered by a
            subscriber requesting an on-demand synchronization.";
    }
    anydata datastore-changes {
        description
            "This contains datastore contents that has changed since the
            previous update, per the terms of the subscription. Changes
            are encoded analogous to the syntax of a corresponding yang-
            patch operation, i.e. a yang-patch operation applied to the
            YANG datastore implied by the previous update to result in the
            current state (and assuming yang-patch could also be applied to
            operational data).";
    }
}
```

```
}
augment "/sn:subscription-started" {
  description
    "This augmentation adds push subscription parameters to the
    notification that a subscription has started and data updates are
    beginning to be sent. This notification shall only be sent to
    receivers of a subscription; it does not constitute a general-
    purpose notification.";
  uses update-policy;
  uses update-qos;
}
augment "/sn:subscription-started/sn:filter-type" {
  description
    "This augmentation allows to include additional update filters
    options to be included as part of the notification that a
    subscription has started.";
  case update-filter {
    description
      "Additional filter options for push subscription.";
    uses update-filter;
  }
}
augment "/sn:subscription-modified" {
  description
    "This augmentation adds push subscription parameters to the
    notification that a subscription has been modified. This
    notification shall only be sent to receivers of a subscription;
    it does not constitute a general-purpose notification.";
  uses update-policy;
  uses update-qos;
}
augment "/sn:subscription-modified/sn:filter-type" {
  description
    "This augmentation allows to include additional update
    filters options to be included as part of the notification
    that a subscription has been modified.";
  case update-filter {
    description
      "Additional filter options for push subscription.";
    uses update-filter;
  }
}
augment "/sn:filters/sn:filter/" +
  "sn:filter-type" {
  description
    "This container adds additional update filter options to the list
    of configurable filters that can be applied to subscriptions.
    This facilitates the reuse of complex filters once defined.";
```

```
    case update-filter {
      uses update-filter;
    }
  }
  augment "/sn:subscription-config/sn:subscription" {
    description
      "Contains the list of subscriptions that are configured,
       as opposed to established via RPC or other means.";
    uses update-policy;
    uses update-qos;
  }
  augment "/sn:subscription-config/sn:subscription/" +
    "sn:filter-type" {
    description
      "Add push filters to selection of filter types.";
    case update-filter {
      uses update-filter;
    }
  }
  augment "/sn:subscriptions/sn:subscription" {
    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, e.g. establish-subscription, delete-subscription,
       and modify-subscription, as well as subscriptions that
       have been established via configuration.";
    uses update-policy;
    uses update-qos;
  }
  augment "/sn:subscriptions/sn:subscription/" +
    "sn:filter-type" {
    description
      "Add push filters to selection of filter types.";
    case update-filter {
      description
        "Additional filter options for push subscription.";
      uses update-filter;
    }
  }
}

<CODE ENDS>
```

## 6. Security Considerations

Subscriptions could be used to attempt to overload publishers of YANG datastores. For this reason, it is important that the publisher has the ability to decline a subscription request if it would deplete its resources. In addition, a publisher needs to be able to suspend an existing subscription when needed. When this occurs, the subscription status is updated accordingly and the receivers are notified. Likewise, requests for subscriptions need to be properly authorized.

A subscription could be used to retrieve data in subtrees that a receiver has no authorized access to. Therefore it is important that data pushed based on subscriptions is authorized in the same way that regular data retrieval operations are. Data being pushed to a receiver needs therefore to be filtered accordingly, just like if the data were being retrieved on-demand. The Netconf Authorization Control Model applies.

A subscription could be configured on another receiver's behalf, with the goal of flooding that receiver with updates. One or more publishers could be used to overwhelm a receiver which doesn't even support subscriptions. Receivers which do not want pushed data need only terminate or refuse any transport sessions from the publisher. In addition, the Netconf Authorization Control Model SHOULD be used to control and restrict authorization of subscription configuration.

For both configured and dynamic subscriptions it is essential to authenticate and authorize that receiver via some transport level mechanism before sending any push updates.

## 7. Acknowledgments

For their valuable comments, discussions, and feedback, we wish to acknowledge Tim Jenkins, Kent Watsen, Susan Hares, Yang Geng, Peipei Guo, Michael Scharf, Sharon Chisholm, and Guangying Zheng.

## 8. References

### 8.1. Normative References

- [I-D:netconf-sub-notif]  
Clemm, A., Gonzalez Prieto, A., Voit, E., Tripathy, A.,  
and E. Nilsen-Nygaard, "Subscribing to YANG-Defined Event  
Notifications", draft-ietf-netconf-subscribed-  
notifications-00 (work in progress), February 2017.



- [RFC6470] Bierman, A., "Network Configuration Protocol (NETCONF) Base Notifications", RFC 6470, DOI 10.17487/RFC6470, February 2012, <<http://www.rfc-editor.org/info/rfc6470>>.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <<http://www.rfc-editor.org/info/rfc6536>>.
- [RFC7895] Bierman, A., Bjorklund, M., and K. Watsen, "YANG Module Library", RFC 7895, DOI 10.17487/RFC7895, June 2016, <<http://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, <<http://www.rfc-editor.org/info/rfc7950>>.
- [RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <<http://www.rfc-editor.org/info/rfc7951>>.
- [RFC8072] Bierman, A., Bjorklund, M., and K. Watsen, "YANG Patch Media Type", RFC 8072, DOI 10.17487/RFC8072, February 2017, <<http://www.rfc-editor.org/info/rfc8072>>.

## 8.2. Informative References

- [RFC1157] Case, J., "A Simple Network Management Protocol (SNMP)", RFC 1157, May 1990.
- [RFC5277] Chisholm, S. and H. Trevino, "NETCONF Event Notifications", RFC 5277, July 2008.
- [RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.
- [RFC7923] Voit, E., Clemm, A., and A. Gonzalez Prieto, "Requirements for Subscription to YANG Datastores", RFC 7923, June 2016.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<http://www.rfc-editor.org/info/rfc8040>>.

## Appendix A. Technologies to be considered for future iterations

### A.1. Proxy YANG Subscription when the Subscriber and Receiver are different

The properties of Dynamic and Configured Subscriptions can be combined to enable deployment models where the Subscriber and Receiver are different. Such separation can be useful with some combination of:

- o An operator does not want the subscription to be dependent on the maintenance of transport level keep-alives. (Transport independence provides different scalability characteristics.)
- o There is not a transport session binding, and a transient Subscription needs to survive in an environment where there is unreliable connectivity with the Receiver and/or Subscriber.
- o An operator wants the Publisher to include highly restrictive capacity management and Subscription security mechanisms outside of domain of existing operational or programmatic interfaces.

To build a Proxy Subscription, first the necessary information must be signaled as part of the <establish-subscription>. Using this set of Subscriber provided information; the same process described within section 3 will be followed.

After a successful establishment, if the Subscriber wishes to track the state of Receiver subscriptions, it may choose to place a separate on-change Subscription into the "Subscriptions" subtree of the YANG Datastore on the Publisher.

### A.2. OpState and Filters

Currently there are ongoing discussions to revise the concept of datastores, allowing for proper handling and distinction of intended versus applied configurations and extending the notion of a datastore to operational data. When finalized, the new concept may open up the possibility for new types of subscription filters, for example, targeting specific datastores and targeting (potentially) differences in datatrees across different datastores.

Likewise, it is conceivable that filters are defined that apply to metadata, such as data nodes for which metadata has been defined that meets a certain criteria.

Defining any such subscription filters at this point would be highly speculative in nature. However, it should be noted that

corresponding extensions may be defined in future specifications. Any such extensions will be straightforward to accommodate by introducing a model that defines new filter types, and augmenting the new filter type into the subscription model.

#### A.3. Splitting push updates

Push updates may become fairly large and extend across multiple subsystems in a YANG-Push Server. As a result, it is conceivable to not combine all updates into a single update message, but to split updates into multiple separate update messages. Such splitting could occur along multiple criteria: limiting the number of data nodes contained in a single update, grouping updates by subtree, grouping updates by internal subsystems (e.g., by line card), or grouping them by other criteria.

Splitting updates bears some resemblance to fragmenting packets. In effect, it can be seen as fragmenting update messages at an application level. However, from a transport perspective, splitting of update messages is not required as long as the transport does not impose a size limitation or provides its own fragmentation mechanism if needed. We assume this to be the case for YANG-Push. In the case of NETCONF, RESTCONF, HTTP/2, no limit on message size is imposed. In case of other transports, any message size limitations need to be handled by the corresponding transport mapping.

There may be some scenarios in which splitting updates might still make sense. For example, if updates are collected from multiple independent subsystems, those updates could be sent separately without need for combining. However, if updates were to be split, other issues arise. Examples include indicating the number of updates to the receiver, distinguishing a missed fragment from a missed update, and the ordering with which updates are received. Proper addressing those issues would result in considerable complexity, while resulting in only very limited gains. In addition, if a subscription is found to result in updates that are too large, a publisher can always reject the request for a subscription while the subscriber is always free to break a subscription up into multiple subscriptions.

#### A.4. Potential Subscription Parameters

A possible is the introduction of an additional parameter "changes-only" for periodic subscription. Including this flag would result in sending at the end of each period an update containing only changes since the last update (i.e. a change-update as in the case of an on-change subscription), not a full snapshot of the subscribed

information. Such an option might be interesting in case of data that is largely static and bandwidth-constrained environments.

#### Appendix B. Issues that are currently being worked and resolved

(To be removed by RFC editor prior to publication)

Issue #6: Data plane notifications and layered headers. Specifically how do we want to enable standard header unification and bundle support vs. the data plane notifications currently defined.

#### Appendix C. Changes between revisions

(To be removed by RFC editor prior to publication)

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. synch-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  
Cisco Systems

Email: albertgo@cisco.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  
Internet-Draft  
Intended status: Standards Track  
Expires: November 22, 2019

A. Clemm  
Futurewei  
E. Voit  
Cisco Systems  
May 21, 2019

Subscription to YANG Datastores  
draft-ietf-netconf-yang-push-25

Abstract

This document describes a mechanism that allows subscriber applications to request a continuous and 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.

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 November 22, 2019.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

## Table of Contents

1. Introduction . . . . .	3
2. Definitions and Acronyms . . . . .	3
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 . . . . .	24
4.3. YANG Notifications . . . . .	25
4.4. YANG RPCs . . . . .	26
5. YANG Module . . . . .	31
6. IANA Considerations . . . . .	48
7. Security Considerations . . . . .	48
8. Acknowledgments . . . . .	50
9. Contributors . . . . .	50
10. References . . . . .	50
10.1. Normative References . . . . .	50
10.2. Informative References . . . . .	51
Appendix A. Appendix A: Subscription Errors . . . . .	52
A.1. RPC Failures . . . . .	52
A.2. Notifications of Failure . . . . .	54



Appendix B. Changes Between Revisions . . . . .	54
Authors' Addresses . . . . .	58

## 1. Introduction

Traditional approaches to provide visibility into managed entities from a remote system 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:

- o Polling incurs significant latency. This latency prohibits many types of application.
- o Polling cycles may be missed and requests may be delayed or get lost, often when the network is under stress and the need for the data is the greatest.
- o Polling requests may undergo slight fluctuations, resulting in intervals of different lengths. The resulting data is difficult to calibrate and compare.
- o 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

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

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

This document uses the terminology defined in [RFC7950], [RFC8341], [RFC8342], and [I-D.draft-ietf-netconf-subscribed-notifications]. In addition, the following terms are introduced:

- o 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"
- o 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.
- o Datastore subscription: A subscription to a stream of datastore node updates.
- o Datastore subtree: A datastore node and all its descendant datastore nodes
- o On-change subscription: A datastore subscription with updates that are triggered when changes in subscribed datastore nodes are detected.
- o Periodic subscription: A datastore subscription with updates that are triggered periodically according to some time interval.
- o Selection filter: Evaluation and/or selection criteria, which may be applied against a targeted set of objects.
- o 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".
- o 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.

The solution builds on [I-D.draft-ietf-netconf-subscribed-notifications]. As defined there, any loss of underlying transport connection will be detected and result in subscription termination (in case of dynamic subscriptions) or suspension (in case of configured subscriptions), ensuring that situations will not occur in which the loss of update notifications would go unnoticed.

### 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:

- o 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.
- o xpath: An "xpath" selection filter is an XPath expression that returns a node set. (XPath is a query language for selecting nodes in an XML document.) 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?           centiseconds
    +--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-datastore-error-info
    +--ro reason?                identityref
    +--ro period-hint?           centiseconds
    +--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.

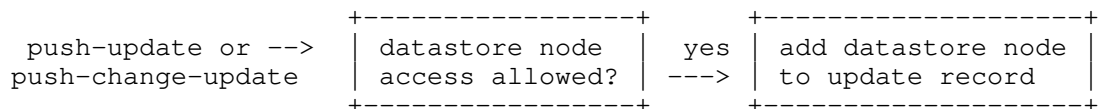


Figure 5: Updated [RFC8341] access control for push updates

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 lifecycle 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. One solution might involve making discoverable to clients which objects are on-change notifiable, specified using another YANG data model. Such a solution is specified in [I-D.draft-ietf-netconf-notification-capabilities]. Until this solution is standardized, 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.

On the receiver side, what action to take when a record with an incomplete-update flag is received depends on the application. It could simply choose to wait and do nothing. It could choose to resynch, actively retrieving all subscribed information. It could also choose to tear down the subscription and start a new one, perhaps with a lesser scope that contains less objects.

### 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 figures. The tree diagram that is used 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.

Because the tree diagram is quite large, its depiction is broken up into several figures. The first figure depicts the augmentations that are introduced in module ietf-yang-push to subscription configuration specified in module ietf-subscribed-notifications.



```

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 ...
|   |   +--rw (yp:update-trigger)
|   |   |   +--:(yp:periodic)
|   |   |   |   +--rw yp:periodic!
|   |   |   |   |   +--rw yp:period                centiseconds
|   |   |   |   |   +--rw yp:anchor-time?          yang:date-and-time
|   |   |   +--:(yp:on-change) {on-change}?
|   |   |   |   +--rw yp:on-change!
|   |   |   |   |   +--rw yp:dampening-period?      centiseconds
|   |   |   |   |   +--rw yp:sync-on-start?         boolean
|   |   |   |   +--rw yp:excluded-change*          change-type

```

Figure 6: Model structure: subscription configuration

The next figure depicts the augmentations of module `ietf-yang-push` made to RPCs specified in module `ietf-subscribed-notifications`. Specifically, these augmentations concern the `establish-subscription` and `modify-subscription` RPCs, which are augmented with parameters that are needed to specify datastore push subscriptions.

```

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
  |   |   |   |   |   |   centiseconds
  |   |   |   |   |   +---w yp:anchor-time?
  |   |   |   |   |   |   yang:date-and-time
  |   |   |   |   +---:(yp:on-change) {on-change}?
  |   |   |   |   |   +---w yp:on-change!
  |   |   |   |   |   |   +---w yp:dampening-period?
  |   |   |   |   |   |   |   centiseconds
  |   |   |   |   |   |   +---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: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                                centiseconds
|      |      |      |      |      +---w yp:anchor-time?                    yang:date-and-time
|      |      |      +---:(yp:on-change) {on-change}?
|      |      |      |      +---w yp:on-change!
|      |      |      |      +---w yp:dampening-period?    centiseconds
|      +---x delete-subscription
|      |      ...
|      +---x kill-subscription
|      |      ...

```

YANG-data (for placement into rpc error responses)

...

Figure 7: Model structure: RPCs

The next figure depicts augmentations of module `ietf-yang-push` to the notifications that are specified in module `ietf-subscribed-notifications`. The augmentations allow the inclusion of subscription configuration parameters that are specific to datastore push subscriptions as part of subscription-started and subscription-modified notifications.

```

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                                centiseconds
    |   +--ro yp:anchor-time?                          yang:date-and-time
  +---:(yp:on-change) {on-change}?
    +--ro yp:on-change!
      +--ro yp:dampening-period?                        centiseconds
      +--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          centiseconds
|      |      |      +---ro yp:anchor-time?   yang:date-and-time
|      +---:(yp:on-change) {on-change}?
|      |      +---ro yp:on-change!
|      |      |      +---ro yp:dampening-period? centiseconds
|      |      |      +---ro yp:sync-on-start?  boolean
|      |      |      +---ro yp:excluded-change* change-type
+---n subscription-terminated
|      ...
+---n subscription-suspended
|      ...

```

Figure 8: Model structure: Notifications

The final figure in this section depicts the parts of module `ietf-yang-push` that are not simply augmentations to another module, but that are newly introduced.

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?     centiseconds
|      +---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?     centiseconds
|      +---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?                centiseconds
+--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 comment?        string
|   |   |   +--ro edit* [edit-id]
|   |   |   |   +--ro edit-id      string
|   |   |   |   +--ro operation    enumeration
|   |   |   |   +--ro target       target-resource-offset
|   |   |   |   +--ro point?       target-resource-offset
|   |   |   |   +--ro where?       enumeration
|   |   |   |   +--ro value?       <anydata>
|   |   +--ro incomplete-update?   empty

```

Figure 9: Model structure (non-augmentation portions)

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 are guided by their 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 had 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 10: 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 11: 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 NETCONF, those parameters are defined in [I-D.draft-ietf-netconf-netconf-event-notifications]. For example, for the following NETCONF request:

```
<rpc message-id="101"
  xmlns="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>
  </rpc>
```

Figure 12: Establish-subscription request example 2

a publisher that cannot serve on-change updates but that can serve periodic updates might return the following NETCONF response:

```
<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 13: 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 using NETCONF.

```
<rpc message-id="102"
  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>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>
</rpc>
```

Figure 14: 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 as part of the RPC error response depend on the specific transport that is used to manage the subscription. For NETCONF, those parameters are specified in [I-D.draft-ietf-netconf-netconf-event-notifications].

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:

```
<rpc message-id="103"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <resync-subscription
    xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push"
    <id>1011</id>
  </resync-subscription>
</netconf:rpc>
```

Figure 15: 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 [RFC8525]. 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

This YANG module imports typedefs from [RFC6991], identities from [RFC8342], the YANG-data extension from [RFC8040], and the yang-patch grouping from [RFC8072]. In addition, it imports and augments many definitions from [I-D.draft-ietf-netconf-subscribed-notifications].

```
<CODE BEGINS> file "ietf-yang-push@2019-05-21.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 Media Type";
  }

  organization
    "IETF NETCONF Working Group";
  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.

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

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

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

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

// RFC Ed.: replace XXXX with actual RFC number and remove this
// note.

revision 2019-05-21 {
  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@2019-05-21.yang) to the date of RFC
        publication.
        (3) Please replace the following reference to
        draft-ietf-netconf-yang-push-25 with RFC number when
        published (i.e. RFC xxxx).";
    reference
        "draft-ietf-netconf-yang-push-25";
}

/*
 * 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.

    This type is based on the edit operations defined for YANG
    Patch, with the difference 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.";
reference
    "RFC 8072: YANG Patch Media Type, section 2.5";
}

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.";
}

typedef centiseconds {
    type uint32;
    description
        "A period of time, measured in units of 0.01 seconds.";
}

/*
 * 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 is the set of prefix and namespace pairs for all YANG modules implemented by the server, where the prefix is the YANG module name and the namespace is as defined by the 'namespace' statement in the YANG module.

If the leaf is encoded in XML, all namespace declarations in scope on the 'stream-xpath-filter' leaf element are added to the set of namespace declarations. If a prefix found in the XML is

already present in the set of namespace declarations, the namespace in the XML is used.

- 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 {
    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 centiseconds;
          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 centiseconds;
      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 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 centiseconds;
    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" {
```



```
    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" {
    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 datastore-criteria;
  }
}

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

/*
 * 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 '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-21.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 [RFC8446].

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.)

- o 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.
- o 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.
- o 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).
- o 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.

NACM provides one means to mitigate these threats on the publisher side. In order to address those threats as a subscriber, a subscriber could monitor the subscription configuration for any unexpected changes. For this, it can subscribe to updates to the YANG datastore nodes that represent his datastore subscriptions. As this data volume is small, a paranoid subscriber could even revert to occasional polling to guard against a compromised subscription against subscription configuration updates itself.

## 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. Contributors

Alberto Gonzalez Prieto  
Microsoft  
albgonz@microsoft.com

Ambika Prasad Tripathy  
Cisco Systems  
ambtripa@cisco.com

Einar Nilsen-Nygaard  
Cisco Systems  
einarnn@cisco.com

Andy Bierman  
YumaWorks  
andy@yumaworks.com

Balazs Lengyel  
Ericsson  
balazs.lengyel@ericsson.com

## 10. References

### 10.1. Normative References

- [I-D.draft-ietf-netconf-subscribed-notifications]  
Voit, E., Clemm, A., Gonzalez Prieto, A., Tripathy, A.,  
and E. Nilsen-Nygaard, "Subscription to YANG Event  
Notifications", draft-ietf-netconf-subscribed-  
notifications-24 (work in progress), April 2019.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate  
Requirement Levels", BCP 14, RFC 2119,  
DOI 10.17487/RFC2119, March 1997,  
<<https://www.rfc-editor.org/info/rfc2119>>.
- [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>>.
- [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>>.
- [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>>.
- [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>>.
- [RFC8525] Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", RFC 8525, DOI 10.17487/RFC8525, March 2019, <<https://www.rfc-editor.org/info/rfc8525>>.

## 10.2. Informative References

- [I-D.draft-ietf-netconf-netconf-event-notifications] Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Dynamic subscription to YANG Events and Datastores over NETCONF", April 2019.

- [I-D.draft-ietf-netconf-notification-capabilities]  
Lengyel, B. and A. Clemm, "YangPush Notification Capabilities", March 2019.
- [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>>.
- [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. 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 in the ietf-subscribed-notifications YANG model or the ietf-yang-push YANG model may be returned as part of the error responses resulting from failed attempts at datastore subscription. For errors defined as part of ietf-subscribed-notifications, please refer to

[I-D.draft-ietf-netconf-subscribed-notifications]. The errors introduced in this document, grouped per RPC, are as follows:

establish-subscription	modify-subscription
-----	-----
cant-exclude	period-unsupported
datastore-not-subscribable	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

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 "modify-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, a number of errors can be returned as part of the corresponding subscription state change notification. For this purpose, the following error identities have been introduced in this document, in addition to those that were already defined in [I-D.draft-ietf-netconf-subscribed-notifications]:

subscription-terminated -----	subscription-suspended -----
datastore-not-subscribable unchanging-selection	period-unsupported update-too-big synchronization-size

## Appendix B. Changes Between Revisions

(To be removed by RFC editor prior to publication)

v24 - v25

- o Minor updates to address IESG review comment regarding referencing the draft which addresses the notification capabilities problem.

v23 - v24

- o Minor updates to address IESG review comments. Moving five of the coauthors to contributors as requested.

v22 - v23

- o Minor updates to address IESG review comments.

v21 - v22

- o Minor updates per Martin Bjorklund's YANG doctor review.

v20 - v21

- o Minor updates, simplifying RPC input conditions.

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

- o 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

- o Minor text fixes.

v12 - v13

- o Hint negotiation models now show error examples.
- o yang-data structures for rpc errors.

v11 - v12

- o Included Martin's review clarifications.
- o QoS moved to subscribed-notifications
- o time-of-update removed as it is redundant with RFC5277's eventTime, and other times from notification-messages.

- o Error model moved to match existing implementations
- o On-change notifiable removed, how to do this is implementation specific.
- o NMDA model supported. Non NMDA version at <https://github.com/netconf-wg/yang-push/>

## v10 - v11

- o Promise model reference added.
- o Error added for no-such-datastore
- o Inherited changes from subscribed notifications (such as optional feature definitions).
- o scrubbed the examples for proper encodings

## v09 - v10

- o 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 appendices, 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 describing the resync subscription RPC.
- o Removed relationships to other drafts and future technology appendices 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  
Futurewei

Email: ludwig@clemm.org

Eric Voit  
Cisco Systems

Email: evoit@cisco.com



NETCONF Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: September 14, 2017

K. Watsen  
Juniper Networks  
M. Abrahamsson  
T-Systems  
March 13, 2017

Zero Touch Provisioning for NETCONF or RESTCONF based Management  
draft-ietf-netconf-zerotouch-13

Abstract

This draft presents a secure technique for establishing a NETCONF or RESTCONF connection between a newly deployed device, configured with just its factory default settings, and its deployment specific network management system (NMS).

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. Please note that no other RFC Editor instructions are specified anywhere else 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-netconf-client-server
- o I-D.ietf-anima-bootstrapping-keyinfra

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 "2017-03-13" --> the publication date of this draft

The following one 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 <http://datatracker.ietf.org/drafts/current/>.

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

This Internet-Draft will expire on September 14, 2017.

## Copyright Notice

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

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

## Table of Contents

1. Introduction . . . . .	4
1.1. Use Cases . . . . .	4
1.2. Terminology . . . . .	5
1.3. Requirements Language . . . . .	6
1.4. Tree Diagram Notation . . . . .	6
2. Guiding Principles . . . . .	7
2.1. Trust Anchors . . . . .	7
2.2. Conveying Trust . . . . .	7
2.3. Conveying Ownership . . . . .	8
3. Types of Zero Touch Information . . . . .	8
3.1. Redirect Information . . . . .	8
3.2. Bootstrap Information . . . . .	9
4. Artifacts . . . . .	10
4.1. Zero Touch Information . . . . .	10

4.2. Owner Certificate . . . . .	11
4.3. Ownership Voucher . . . . .	12
5. Artifact Groupings . . . . .	12
5.1. Unsigned Information . . . . .	12
5.2. Signed Information (without Revocations) . . . . .	13
5.3. Signed Information (with Revocations) . . . . .	13
6. Sources of Bootstrapping Data . . . . .	14
6.1. Removable Storage . . . . .	14
6.2. DNS Server . . . . .	15
6.3. DHCP Server . . . . .	16
6.4. Bootstrap Server . . . . .	17
7. Workflow Overview . . . . .	18
7.1. Onboarding and Ordering Devices . . . . .	19
7.2. Owner Stages the Network for Bootstrap . . . . .	21
7.3. Device Powers On . . . . .	23
8. Device Details . . . . .	25
8.1. Factory Default State . . . . .	25
8.2. Boot Sequence . . . . .	26
8.3. Processing a Source of Bootstrapping Data . . . . .	27
8.4. Validating Signed Data . . . . .	28
8.5. Processing Redirect Information . . . . .	29
8.6. Processing Bootstrap Information . . . . .	30
9. The Zero Touch Information Artifact . . . . .	31
9.1. Tree Diagram . . . . .	31
9.2. Example Usage . . . . .	31
9.3. YANG Module . . . . .	34
10. The Zero Touch Bootstrap Server API . . . . .	39
10.1. Tree Diagram . . . . .	39
10.2. Example Usage . . . . .	40
10.3. YANG Module . . . . .	43
11. Security Considerations . . . . .	51
11.1. Immutable storage for trust anchors . . . . .	51
11.2. Clock Sensitivity . . . . .	51
11.3. Blindly authenticating a bootstrap server . . . . .	51
11.4. Entropy loss over time . . . . .	52
11.5. Serial Numbers . . . . .	52
11.6. Sequencing Sources of Bootstrapping Data . . . . .	52
12. IANA Considerations . . . . .	52
12.1. The BOOTP Manufacturer Extensions and DHCP Options Registry . . . . .	52
12.2. The IETF XML Registry . . . . .	53
12.3. The YANG Module Names Registry . . . . .	54
13. Other Considerations . . . . .	54
14. Acknowledgements . . . . .	54
15. References . . . . .	54
15.1. Normative References . . . . .	54
15.2. Informative References . . . . .	56
Appendix A. Change Log . . . . .	58

A.1.	ID to 00	. . . . .	58
A.2.	00 to 01	. . . . .	58
A.3.	01 to 02	. . . . .	58
A.4.	02 to 03	. . . . .	59
A.5.	03 to 04	. . . . .	59
A.6.	04 to 05	. . . . .	59
A.7.	05 to 06	. . . . .	60
A.8.	06 to 07	. . . . .	60
A.9.	07 to 08	. . . . .	60
A.10.	08 to 09	. . . . .	60
A.11.	09 to 10	. . . . .	60
A.12.	10 to 11	. . . . .	61
A.13.	11 to 12	. . . . .	61
A.14.	12 to 13	. . . . .	61
Authors' Addresses			62

## 1. Introduction

A fundamental business requirement for any network operator is to reduce costs where possible. For network operators, deploying devices to many locations can be a significant cost, as sending trained specialists to each site to do installations is both cost prohibitive and does not scale.

This document defines a bootstrapping strategy enabling devices to securely obtain bootstrapping data with no installer input, beyond physical placement and connecting network and power cables. The ultimate goal of this document is to enable a secure NETCONF [RFC6241] or RESTCONF [RFC8040] connection to the deployment specific network management system (NMS).

### 1.1. Use Cases

#### o Device connecting to a remotely administered network

This use-case involves scenarios, such as a remote branch office or convenience store, whereby a device connects as an access gateway to an ISP's network. Assuming it is not possible to customize the ISP's network to provide any bootstrapping support, and with no other nearby device to leverage, the device has no recourse but to reach out to an Internet-based bootstrap server to bootstrap off of.

#### o Device connecting to a locally administered network

This use-case covers all other scenarios and differs only in that the device may additionally leverage nearby devices, which may direct it to use a local service to bootstrap off of. If

no such information is available, or the device is unable to use the information provided, it can then reach out to network just as it would for the remotely administered network use-case.

## 1.2. Terminology

This document uses the following terms:

**Artifact:** The term "artifact" is used throughout to represent the any of the three artifacts defined in Section 4. These artifacts collectively provide all the bootstrapping data a device needs.

**Bootstrapping Data:** The term "bootstrapping data" is used throughout this document to refer to the collection of data that a device may obtain from any source of bootstrapping data. Specifically, it refers to the artifacts defined in Section 4.

**Bootstrap Information:** The term "bootstrap information" is used herein to refer to one of the bootstrapping artifacts defined in Section 4. Specifically, bootstrap information is the bootstrapping data that guides a device to, for instance, install a specific boot-image and commit a specific configuration.

**Bootstrap Server:** The term "bootstrap server" is used within this document to mean any RESTCONF server implementing the YANG module defined in Section 10.3.

**Device:** The term "device" is used throughout this document to refer to the network element that needs to be bootstrapped. See Section 8 for more information about devices.

**Initial Secure Device Identifier (IDevID):** The term "IDevID" is defined in [Std-802.1AR-2009] as the secure device identifier (DevID) installed on the device by the manufacturer. This identifier is used in this document to enable a Bootstrap Server to securely identify and authenticate a device.

**Manufacturer:** The term "manufacturer" is used herein to refer to the manufacturer of a device or a delegate of the manufacturer.

**Network Management System (NMS):** The acronym "NMS" is used throughout this document to refer to the deployment specific management system that the bootstrapping process is responsible for introducing devices to. From a device's perspective, when the bootstrapping process has completed, the NMS is a NETCONF or RESTCONF client.

Owner: See Rightful Owner.

Redirect Information: The term "bootstrap information" is used herein to refer to one of the bootstrapping artifacts defined in Section 4. Specifically, redirect information is the bootstrapping data that directs a device to connect to a bootstrap server.

Redirect Server: The term "redirect server" is used to refer to a subset of bootstrap servers that only returns redirect information. A redirect server is particularly useful when hosted by a manufacturer, to redirect devices to deployment-specific bootstrap servers.

Rightful Owner: The term "rightful owner" is used herein to refer to the person or organization that purchased or otherwise owns a device. Ownership is further described in Section 2.3.

Signed Data: The term "signed data" is used throughout to mean either redirect information or bootstrap information that has been signed by a device's rightful owner's private key.

Unsigned Data: The term "unsigned data" is used throughout to mean either redirect information or bootstrap information that has not been signed by a device's rightful owner's private key.

### 1.3. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in the sections below are to be interpreted as described in RFC 2119 [RFC2119].

### 1.4. Tree Diagram Notation

A simplified graphical representation of the data models is used in this document. The meaning of the symbols in these diagrams is as follows:

- o Brackets "[" and "]" enclose list keys.
- o Braces "{" and "}" enclose feature names, and indicate that the named feature must be present for the subtree to be present.
- o Abbreviations before data node names: "rw" (read-write) represents configuration data and "ro" (read-only) represents state data.

- o Symbols after data node names: "?" means an optional node, "!" means a presence container, and "\*" denotes a list and leaf-list.
- o Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- o Ellipsis ("...") stands for contents of subtrees that are not shown.

## 2. Guiding Principles

This section provides overarching principles guiding the solution presented in this document.

### 2.1. Trust Anchors

A trust anchor is used in cryptography to represent an entity in which trust is implicit and not derived. In public key infrastructure using X.509 certificates, a root certificate is the trust anchor, from which a chain of trust is derived. The solution presented in this document requires that all the entities involved (e.g., devices, bootstrap servers, NMSs) possess specific trust anchors in order to ensure mutual authentication throughout the zero touch bootstrapping process.

### 2.2. Conveying Trust

A device in its factory default state possesses a limited set of manufacturer specified trust anchors. In this document, there are two types of trust anchors of interest. The first type of trust anchor is used to authenticate a secure (e.g., HTTPS) connection to, for instance, a manufacturer-hosted Internet-based bootstrap server. The second type of trust anchor is used to authenticate manufacturer-signed data, such as the ownership voucher artifact described in Section 4.3.

Using the first type of trust anchor, trust is conveyed by the device first authenticating the server (e.g., a bootstrap server), and then by the device trusting that the server would only provide data that its rightful owner staged for it to find. Thereby the device can trust any information returned from the server.

Using the second type of trust anchor, trust is conveyed by the device first authenticating that an artifact has been signed by its rightful owner, and thereby can trust any information held within the artifact.

Notably, redirect information, as described in Section 3.1, may include more trust anchors, which illustrates another way in which trust can be conveyed.

### 2.3. Conveying Ownership

The ultimate goal of this document is to enable a device to establish a secure connection with its rightful owner's NMS. This entails the manufacturer being able to track who is the rightful owner of a device (not defined in this document), as well as an ability to convey that information to devices (defined in this document).

Matching the two ways to convey trust (Section 2.2), this document provides two ways to convey ownership, by using a trusted bootstrap server (Section 6.4) or by using an ownership voucher (Section 4.3).

When a device connects to a trusted bootstrap server, one that was preconfigured into its factory default configuration, it implicitly trusts that the bootstrap server would only provide data that its rightful owner staged for it to find. That is, ownership is conveyed by the administrator of the bootstrap server (e.g., a manufacturer) taking the onus of ensuring that only data configured by a device's rightful owner is made available to the device. With this approach, the assignment of a device to an owner is ephemeral, as the administrator can reassign a device to another owner at any time.

When a device is presented signed bootstrapping data, it can authenticate that its rightful owner provided the data by verifying the signature over the data using an additional artifact defined within this document, the ownership voucher. With this approach, ownership is conveyed by the manufacturer (or delegate) taking the onus of ensuring that the ownership vouchers it issues are accurate.

## 3. Types of Zero Touch Information

This document defines two types of information that devices access during the bootstrapping process. These information types are described in this section.

### 3.1. Redirect Information

Redirect information provides information to redirect a device to a bootstrap server. Redirect information encodes a list of bootstrap servers, each defined by its hostname or IP address, an optional port, and an optional trust anchor certificate.

Redirect information is YANG modeled data formally defined by the "redirect-information" grouping in the YANG module presented in



Section 9.3. This grouping has the tree diagram shown below. Please see Section 1.4 for tree diagram notation.

```
+--:(redirect-information)
  +--ro redirect-information
    +--ro bootstrap-server* [address]
      +--ro address          inet:host
      +--ro port?            inet:port-number
      +--ro trust-anchor?    binary
```

Redirect information MAY be trusted or untrusted. The redirect information is trusted whenever it is obtained via a secure connection to a trusted bootstrap server, or whenever it is signed by the device's rightful owner. In all other cases, the redirect information is untrusted.

Trusted redirect information is useful for enabling a device to establish a secure connection to a bootstrap server, which is possible when the redirect information includes the bootstrap server's trust anchor certificate. When a device is able to establish a secure connection to a bootstrap server, the bootstrapping data does not have to be signed in order to be trusted, as described in Section 2.2.

Untrusted redirect information is useful for directing a device to a bootstrap server where signed data has been staged for it to obtain. When the redirect information is untrusted, the device MUST discard any potentially included trust anchor certificates. When the redirect information is untrusted, a device MAY establish a provisional connection to any of the specified bootstrap servers. A provisional connection is accomplished by the device blindly accepting the bootstrap server's TLS certificate. In this case, the device MUST NOT trust the bootstrap server, and data provided by the bootstrap server MUST be signed for it to be of any use to the device.

How devices process redirect information is described more formally in Section 8.5.

### 3.2. Bootstrap Information

Bootstrap information provides all the data necessary for a device to bootstrap itself, in order to be considered ready to be managed (e.g., by an NMS). As defined in this document, this data includes information about a boot image the device MUST be running, an initial configuration the device MUST commit, and optional scripts that, if specified, the device MUST successfully execute.

Bootstrap information is YANG modeled data formally defined by the "bootstrap-information" grouping in the YANG module presented in Section 9.3. This grouping has the tree diagram shown below. Please see Section 1.4 for tree diagram notation.

```

+---:(bootstrap-information)
  +--ro bootstrap-information
    +--ro boot-image
      |   +--ro name          string
      |   +--ro (hash-algorithm)
      |   |   +---:(sha256)
      |   |   |   +--ro sha256?    string
      |   |   +--ro uri*         inet:uri
      +--ro configuration-handling    enumeration
      +--ro pre-configuration-script?  script
      +--ro configuration?
      +--ro post-configuration-script?  script
```

Bootstrap information MUST be trusted for it to be of any use to a device. There is no option for a device to process untrusted bootstrap information.

Bootstrap information is trusted whenever it is obtained via a secure connection to a trusted bootstrap server, or whenever it is signed by the device's rightful owner. In all other cases, the bootstrap information is untrusted.

How devices process bootstrap information is described more formally in Section 8.6.

## 4. Artifacts

This document defines three artifacts that can be made available to devices while they are bootstrapping. As will be seen in Section 6, each source of bootstrapping information specifies a means for providing each of the artifacts defined in this section.

### 4.1. Zero Touch Information

The information artifact encodes the essential bootstrapping data for the device. This artifact is used to encode the redirect information and bootstrap information types discussed in Section 3.

The information artifact is a PKCS#7 SignedData structure, as specified by Section 9.1 of [RFC2315], encoded using ASN.1 distinguished encoding rules (DER), as specified in ITU-T X.690.

Regardless how the information artifact is conveyed, the PKCS#7 structure MUST contain JSON-encoded content conforming to the YANG module specified in Section 9.3.

When the information artifact is conveyed over an untrusted transport (Section 2.2), the PKCS#7 structure MUST also contain a 'signerInfo' structure, as described in Section 9.1 of [RFC2315], containing a signature generated over the content using the private key associated with the owner certificate (Section 4.2).

#### 4.2. Owner Certificate

The owner certificate artifact is a certificate that is used to identify an 'owner' (e.g., an organization), as known to a trusted certificate authority. The owner certificate is signed by a trusted certificate authority (CA), whose certificate is placed into the ownership voucher (Section 4.3).

The owner certificate is used by a device to verify the signature attached to the information artifact (Section 4.1) that the device SHOULD have also received, as described in Section 5. In particular, the device verifies signature using the public key in the owner certificate over the content contained within the information artifact.

In order to validate the owner certificate, a device MUST verify that the owner certificate's certificate chain includes the certificate specified by the ownership voucher (Section 4.3) that the device SHOULD have also received, as described in Section 5, and the device MUST verify that owner certificate contains an identifier matching the one specified in the voucher and, for devices that insist on verifying certificate revocation status, the device MUST verify that the certificate has neither expired nor been revoked.

The owner certificate artifact is formally an unsigned PKCS #7 SignedData structure as specified by Section 9.1 in [RFC2315], encoded using ASN.1 distinguished encoding rules (DER), as specified in ITU-T X.690.

The owner certificate artifact MUST contain the owner certificate itself and all intermediate certificates leading up to the trust anchor certificate specified in the ownership voucher. The owner certificate artifact MAY optionally include the trust anchor certificate.

Additionally, if needed by the device, the owner certificate artifact MAY also contain suitably fresh CRLs [RFC5280] and/or OCSP Responses [RFC6960].

#### 4.3. Ownership Voucher

The ownership voucher artifact is used to securely identify a device's owner, as it is known to the manufacturer. The ownership voucher is signed by the device's manufacturer or delegate.

The ownership voucher is used by a device to verify the owner certificate (Section 4.2) that the device SHOULD have also received, as described in Section 5. In particular, the device verifies that the owner certificate's chain of trust includes the trusted certificate included in the voucher, and the device also verifies that the owner certificate contains an identifier matching the one specified in the voucher.

In order to validate the voucher, a device MUST verify that the voucher was signed by the private key associated with a trusted certificate known to the device in its factory default state, as described in Section 8.1, and the device MUST verify that the voucher includes the device's unique identifier (e.g., serial number) and, if the voucher contains an expiration date, the device MUST also verify that the voucher has not expired.

The ownership voucher artifact, including its encoding, is formally defined in [I-D.ietf-anima-voucher].

#### 5. Artifact Groupings

Section 4 lists all the possible bootstrapping artifacts, but only certain groupings of these artifacts make sense to return in the various bootstrapping situations described in this document. The remainder of this section identifies these groupings to further clarify how the artifacts are used.

##### 5.1. Unsigned Information

The first grouping of artifacts is for unsigned information. That is, when the information artifact (Section 4.1) has not been signed.

Unsigned information is useful for cases when transport level security can be used to convey trust (e.g., HTTPS), or when the information can be processed in a provisional manner (i.e. unsigned redirect information).

Conveying unsigned information entails communicating just one of the three artifacts listed in Section 4 as follows:

List of artifacts included in this grouping:

- zero touch information (with no embedded signature)

## 5.2. Signed Information (without Revocations)

The second grouping of artifacts is for when the information artifact (Section 4.1) has been signed, without any revocation information.

Signed information is needed when the information is obtained from an untrusted source of bootstrapping data (Section 6) and yet it is desired that the device be able to trust the information (i.e. no provisional processing).

Revocation information may not need to be provided because, for instance, the device only uses revocation information obtained dynamically from Internet based resources. Another possible reason may be because the device does not have a reliable clock, and therefore the manufacturer decides to never revoke information (e.g., ownership assignments are forever).

Conveying signed information without revocation information entails communicating all three of the artifacts listed in Section 4 as follows:

List of artifacts included in this grouping:

- zero touch information (with an embedded signature)
- owner certificate (with no revocation structures)
- ownership voucher

## 5.3. Signed Information (with Revocations)

The third grouping of artifacts is for when the information artifact (Section 4.1) has been signed and also includes revocation information.

Signed information, as described above, is needed when the information is obtained from an untrusted source of bootstrapping data (Section 6) and yet it is desired that the device be able to trust the information (i.e. no provisional processing).

Revocation information may need to be provided because, for instance, the device insists on being able to verify revocations and the device is deployed on a private network and therefore unable to obtain the revocation information from Internet based resources.

Conveying signed information with revocation information entails communicating all three of the artifacts listed in Section 4 as follows:

List of artifacts included in this grouping:

- zero touch information (with an embedded signature)
- owner certificate (with revocation structures)
- ownership voucher

## 6. Sources of Bootstrapping Data

This section defines some sources for zero touch bootstrapping data that a device can access. The list of sources defined here is not meant to be exhaustive. It is left to future documents to define additional sources for obtaining zero touch bootstrapping data.

For each source defined in this section, details are given for how each of the three artifacts listed in Section 4 is provided.

### 6.1. Removable Storage

A directly attached removable storage device (e.g., a USB flash drive) MAY be used as a source of zero touch bootstrapping data.

To use a removable storage device as a source of bootstrapping data, a device need only detect if the removable storage device is plugged in and mount its filesystem.

Use of a removable storage device is compelling, as it doesn't require any external infrastructure to work. It is also compelling that the raw boot image file can be located on the removable storage device, enabling a removable storage device to be a fully self-standing bootstrapping solution.

A removable storage device is an untrusted source of bootstrapping data. This means that the information stored on the removable storage device either MUST be signed, or it MUST be information that can be processed provisionally (e.g., unsigned redirect information).

From an artifact perspective, since a removable storage device presents itself as a filesystem, the bootstrapping artifacts need to be presented as files. The three artifacts defined in Section 4 are mapped to files below.

Artifact to File Mapping:

Information: Mapped to a file containing the binary artifact described in Section 4.1.

Owner Certificate: Mapped to a file containing the binary artifact described in Section 4.2.

Ownership Voucher: Mapped to a file containing the binary artifact described in Section 4.3.

The format of the removable storage device's filesystem and the naming of the files are outside the scope of this document. However, in order to facilitate interoperability, it is RECOMMENDED devices support open and/or standards based filesystems. It is also RECOMMENDED that devices assume a file naming convention that enables more than one instance of bootstrapping data to exist on a removable storage device. The file naming convention SHOULD be unique to the manufacturer, in order to enable bootstrapping data from multiple manufacturers to exist on a removable storage device.

## 6.2. DNS Server

A DNS server MAY be used as a source of zero touch bootstrapping data.

Using a DNS server may be a compelling option for deployments having existing DNS infrastructure, as it enables a touchless bootstrapping option that does not entail utilizing an Internet based resource hosted by a 3rd-party.

To use a DNS server as a source of bootstrapping data, a device MAY perform a multicast DNS [RFC6762] query searching for the service "\_zerotouch.\_tcp.local.". Alternatively the device MAY perform DNS-SD [RFC6763] via normal DNS operation, using the domain returned to it from the DHCP server; for example, searching for the service "\_zerotouch.\_tcp.example.com".

Unsigned DNS records (not using DNSSEC as described in [RFC6698]) are an untrusted source of bootstrapping data. This means that the information stored in the DNS records either MUST be signed, or it MUST be information that can be processed provisionally (e.g., unsigned redirect information).

From an artifact perspective, since a DNS server presents resource records (Section 3.2.1 of [RFC1035]), the bootstrapping artifacts need to be presented as resource records. The three artifacts defined in Section 4 are mapped to resource records below.

### Artifact to Resource Record Mapping:

Information: Mapped to a TXT record called "zt-info" containing the base64-encoding of the binary artifact described in Section 4.1.

Owner Certificate: Mapped to a TXT record called "zt-cert" containing the base64-encoding of the binary artifact described in Section 4.2.

Ownership Voucher: Mapped to a TXT record called "zt-voucher" containing the base64-encoding of the binary artifact described in Section 4.3.

TXT records have an upper size limit of 65535 bytes (Section 3.2.1 in RFC1035), since 'RDLENGTH' is a 16-bit field. Please see Section 3.1.3 in RFC4408 for how a TXT record can achieve this size. Due to this size limitation, some information artifacts may not fit. In particular, the bootstrap information artifact could hit this upper bound, depending on the size of the included configuration and scripts.

When bootstrap information is provided, it is notable that the URL for the boot-image the device can download would have to point to another server (e.g., http://, ftp://, etc.), as DNS servers do not themselves distribute files.

### 6.3. DHCP Server

A DHCP server MAY be used as a source of zero touch bootstrapping data.

To use a DHCP server as a source of bootstrapping data, a device need only send a DHCP lease request to a DHCP server. However, the device SHOULD pass the Vendor Class Identifier (option 60) field in its DHCP lease request, so the DHCP server can return bootstrap information shared by devices from the same vendor. However, if it is desired to return device-specific bootstrap information, then the device SHOULD also send the Client Identifier (option 61) field in its DHCP lease request, so the DHCP server can select the specific bootstrap information that has been staged for that one device.

Using a DHCP server may be a compelling option for deployments having existing DHCP infrastructure, as it enables a touchless bootstrapping option that does not entail utilizing an Internet based resource hosted by a 3rd-party.

A DHCP server is an untrusted source of bootstrapping data. This means that the information returned by the DHCP server either MUST be signed, or it MUST be information that can be processed provisionally (e.g., unsigned redirect information).

From an artifact perspective, since a DHCP server presents data as DHCP options, the bootstrapping artifacts need to be presented as



DHCP options, specifically the ones specified in Section 12.1. The three artifacts defined in Section 4 are mapped to the DHCP options specified in Section 12.1 below.

Artifact to DHCP Option Field Mapping:

Information: Mapped to the DHCP option field "zerotouch-information" containing the binary artifact described in Section 4.1.

Owner Certificate: Mapped to the DHCP option field "owner-certificate" containing the binary artifact described in Section 4.2.

Ownership Voucher: Mapped to the DHCP option field "ownership-voucher" containing the binary artifact described in Section 4.3.

When bootstrap information is provided, it is notable that the URL for the boot-image the device can download would have to point to another server (e.g., http://, ftp://, etc.), as DHCP servers do not themselves distribute files.

#### 6.4. Bootstrap Server

A bootstrap server MAY be used as a source of zero touch bootstrapping data. A bootstrap server is defined as a RESTCONF [RFC8040] server implementing the YANG module provided in Section 10.

Unlike any other source of bootstrap data described in this document, a bootstrap server is not only a source of data, but it can also receive data from devices using the YANG-defined "notification" action statement defined in the YANG module (Section 10.3). The data sent from devices both enables visibility into the bootstrapping process (e.g., warnings and errors) as well as provides potentially useful completion status information (e.g., the device's SSH host-keys).

To use a bootstrap server as a source of bootstrapping data, a device MUST use the RESTCONF protocol to access the YANG container node /device/, passing its own serial number in the URL as the key to the 'device' list.

Using a bootstrap server as a source of bootstrapping data is a compelling option as it uses transport-level security in lieu of signed data, which may be easier to deploy in some situations. Additionally, the bootstrap server is able to receive notifications

from devices, which may be critical to some deployments (e.g., the passing of the device's SSH host keys).

A bootstrap server may be trusted or an untrusted source of bootstrapping data, depending on how the device learned about the bootstrap server's trust anchor from a trusted source. When a bootstrap server is trusted, the information returned from it MAY be signed. However, when the server is untrusted, in order for its information to be of any use to the device, the information MUST either be signed or be information that can be processed provisionally (e.g., unsigned redirect information).

When a device is able to trust a bootstrap server, it MUST send its IDevID certificate in the form of a TLS client certificate, and it MUST send notifications to the bootstrap server. When a device is not able to trust a bootstrap server, it MUST NOT send its IDevID certificate in the form of a TLS client certificate, and it MUST NOT send any notifications to the bootstrap server.

From an artifact perspective, since a bootstrap server presents data as a YANG-modeled data, the bootstrapping artifacts need to be mapped to nodes in the YANG module. The three artifacts defined in Section 4 are mapped to bootstrap server nodes defined in Section 10.3 below.

#### Artifact to Bootstrap Server Node Mapping:

Information: Mapped to the node /device/zerotouch-information.

Owner Certificate: Mapped to the leaf node /device/owner-certificate.

Ownership Voucher: Mapped to the leaf node /device/ownership-voucher.

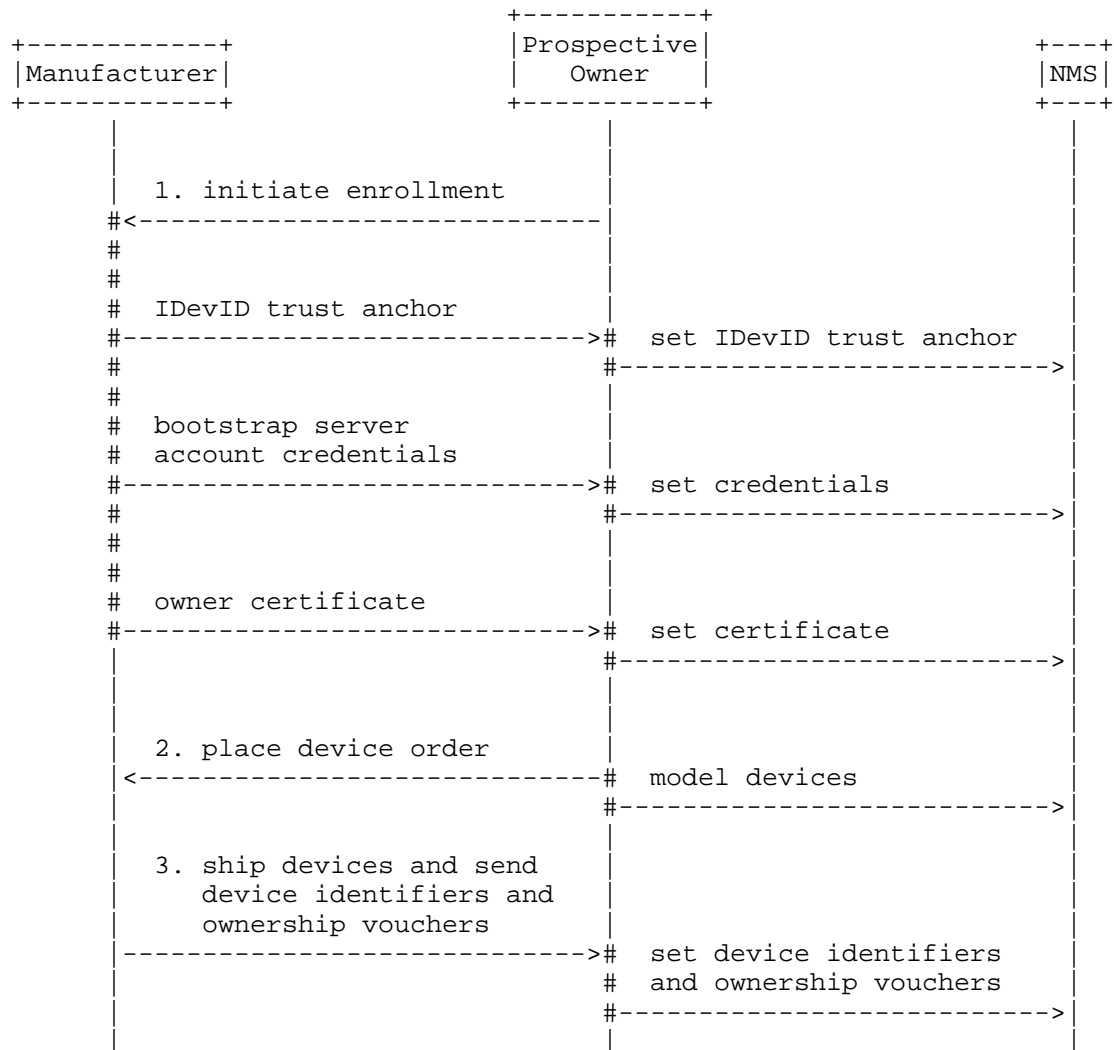
While RESTCONF servers typically support a nested hierarchy of resources, zero touch bootstrap servers only need to support the paths /device and /device/notification. The device processing instructions provided in Section 8.3 only uses these two URLs.

## 7. Workflow Overview

The zero touch solution presented in this document is conceptualized to be composed of the workflows described in this section. Implementations MAY vary in details. Each diagram is followed by a detailed description of the steps presented in the diagram, with further explanation on how implementations may vary.

## 7.1. Onboarding and Ordering Devices

The following diagram illustrates key interactions that may occur from when a prospective owner enrolls in a manufacturer's zero touch program to when the manufacturer ships devices for an order placed by the prospective owner.



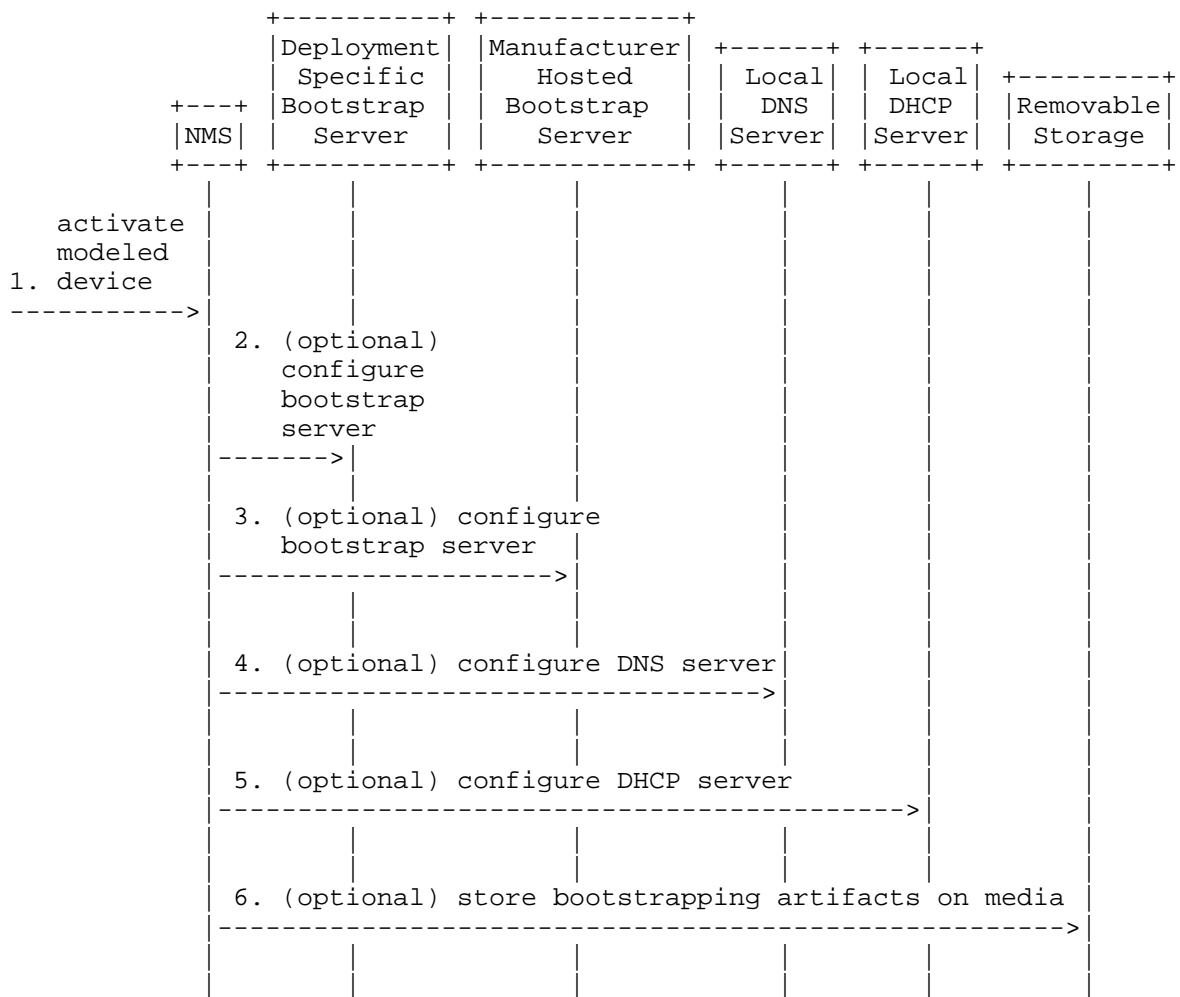
Each numbered item below corresponds to a numbered item in the diagram above.

1. A prospective owner of a manufacturer's devices, or an existing owner that wishes to start using zero touch for future device orders, initiates an enrollment process with the manufacturer or delegate. This process includes the following:
  - \* Regardless how the prospective owner intends to bootstrap their devices, they will always obtain from the manufacturer or delegate the trust anchor certificate for its device's IDevID certificates. This certificate will need to be installed on the prospective owner's NMS so that the NMS can subsequently authenticate the device's IDevID certificates.
  - \* If the manufacturer hosts an Internet based bootstrap server (e.g., a redirect server) such as described in Section 6.4, then credentials necessary to configure the bootstrap server would be provided to the prospective owner. If the bootstrap server is configurable through an API (outside the scope of this document), then the credentials might be installed on the prospective owner's NMS so that the NMS can subsequently configure the manufacturer-hosted bootstrap server directly.
  - \* If the manufacturer's devices are able to validate signed data (Section 8.4), then the manufacturer, acting as a certificate authority, may additionally sign an owner certificate for the prospective owner. Alternatively, and not depicted, the owner may obtain an owner certificate from a manufacturer-trusted 3rd-party certificate authority, and report that certificate to the manufacturer. How the owner certificate is used to enable devices to validate signed bootstrapping data is described in Section 8.4. Assuming the prospective owner's NMS is able to prepare and sign the bootstrapping data, the owner certificate would be installed on the NMS at this time.
2. Some time later, the prospective owner places an order with the manufacturer (or delegate), perhaps with a special flag checked for zero touch handling. At this time, or perhaps before placing the order, the owner may model the devices in their NMS, creating virtual objects for the devices with no real-world device associations. For instance the model can be used to simulate the device's location in the network and the configuration it should have when fully operational.
3. When the manufacturer or delegate fulfills the order, shipping the devices to their intended locations, they may notify the owner of the devices's unique identifiers (e.g., serial numbers) and shipping destinations, which the owner may use to stage the network for when the devices power on. Additionally, the manufacturer may send one or more ownership vouchers,

cryptographically assigning ownership of those devices to the rightful owner. The owner may set this information on their NMS, perhaps binding specific modeled devices to the unique identifiers and ownership vouchers.

## 7.2. Owner Stages the Network for Bootstrap

The following diagram illustrates how an owner might stage the network for bootstrapping devices.



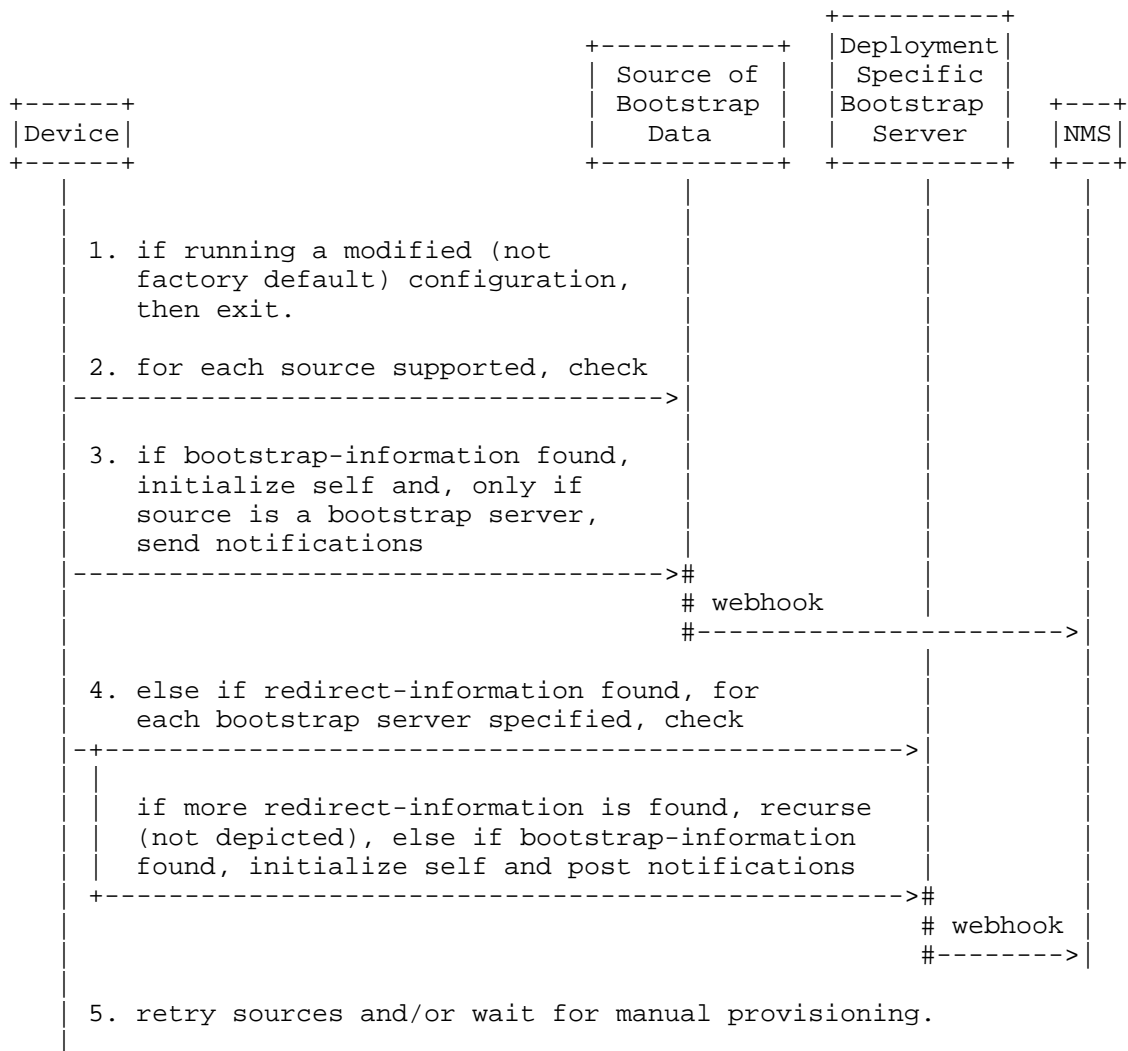
Each numbered item below corresponds to a numbered item in the diagram above.

1. Having previously modeled the devices, including setting their fully operational configurations and associating both device identifiers (e.g., serial numbers) and ownership vouchers, the owner "activates" one or more modeled devices. That is, the owner tells the NMS to perform the steps necessary to prepare for when the real-world devices power up and initiate the bootstrapping process. Note that, in some deployments, this step might be combined with the last step from the previous workflow. Here it is depicted that an NMS performs the steps, but they may be performed manually or through some other mechanism.
2. If it is desired to use a deployment specific bootstrap server, it MUST be configured to provide the bootstrapping information for the specific devices. Configuring the bootstrap server MAY occur via a programmatic API not defined by this document. Illustrated here as an external component, the bootstrap server MAY be implemented as an internal component of the NMS itself.
3. If it is desired to use a manufacturer (or delegate) hosted bootstrap server, it MUST be configured to provide the bootstrapping information for the specific devices. The configuration MUST be either redirect or bootstrap information. That is, either the manufacturer hosted bootstrap server will redirect the device to another bootstrap server, or provide the device with its bootstrapping information itself. The types of bootstrapping information the manufacturer hosted bootstrap server supports MAY vary by implementation; some implementations may only support redirect information, or only support bootstrap information, or support both redirect and bootstrap information. Configuring the bootstrap server MAY occur via a programmatic API not defined by this document.
4. If it is desired to use a DNS server to supply bootstrapping information, a DNS server needs to be configured. If multicast DNS-SD is desired, then the server MUST reside on the local network, otherwise the DNS server MAY reside on a remote network. Please see Section 6.2 for more information about how to configure DNS servers. Configuring the DNS server MAY occur via a programmatic API not defined by this document.
5. If it is desired to use a DHCP server to supply bootstrapping data, a DHCP server needs to be configured. The DHCP server may be accessed directly or via a DHCP relay. Please see Section 6.3 for more information about how to configure DHCP servers. Configuring the DHCP server MAY occur via a programmatic API not defined by this document.

6. If it is desired to use a removable storage device (e.g., USB flash drive) to supply bootstrapping information, the information would need to be placed onto it. Please see Section 6.1 for more information about how to configure a removable storage device.

### 7.3. Device Powers On

The following diagram illustrates the sequence of activities that occur when a device powers on.



The interactions in the above diagram are described below.

1. Upon power being applied, the device's bootstrapping logic first checks to see if it is running in its factory default state. If it is in a modified state, then the bootstrapping logic exits and none of the following interactions occur.
2. For each source of bootstrapping data the device supports, preferably in order of closeness to the device (e.g., removable storage before Internet based servers), the device checks to see if there is any bootstrapping data for it there.
3. If bootstrap-information is found, the device initializes itself accordingly (e.g., installing a boot-image and committing an initial configuration). If the source is a bootstrap server, and the bootstrap server can be trusted (i.e., TLS-level authentication), the device also sends progress notifications to the bootstrap server.
  - \* The contents of the initial configuration SHOULD configure an administrator account on the device (e.g., username, ssh-rsa key, etc.) and SHOULD configure the device either to listen for NETCONF or RESTCONF connections or to initiate call home connections [RFC8071].
  - \* If the bootstrap server supports forwarding device notifications to external systems (e.g., via a webhook), the "bootstrap-complete" notification (Section 10.3) informs the external system to know when it can, for instance, initiate a connection to the device (assuming it knows the device's address and the device was configured to listen for connections). To support this further, the bootstrap-complete notification also relays the device's SSH host keys and/or TLS certificates, with which the external system can use to authenticate subsequent connections to the device.

If the device is ever able to complete the bootstrapping process successfully (i.e., no longer running its factory default configuration), it exits the bootstrapping logic without considering any additional sources of bootstrapping data.

4. Otherwise, if redirect-information is found, the device iterates through the list of specified bootstrap servers, checking to see if there is any bootstrapping data for it on them. If the bootstrap server returns more redirect-information, then the device processes it recursively. Otherwise, if the bootstrap server returns bootstrap-information, the device processes it following the description provided in (3) above.

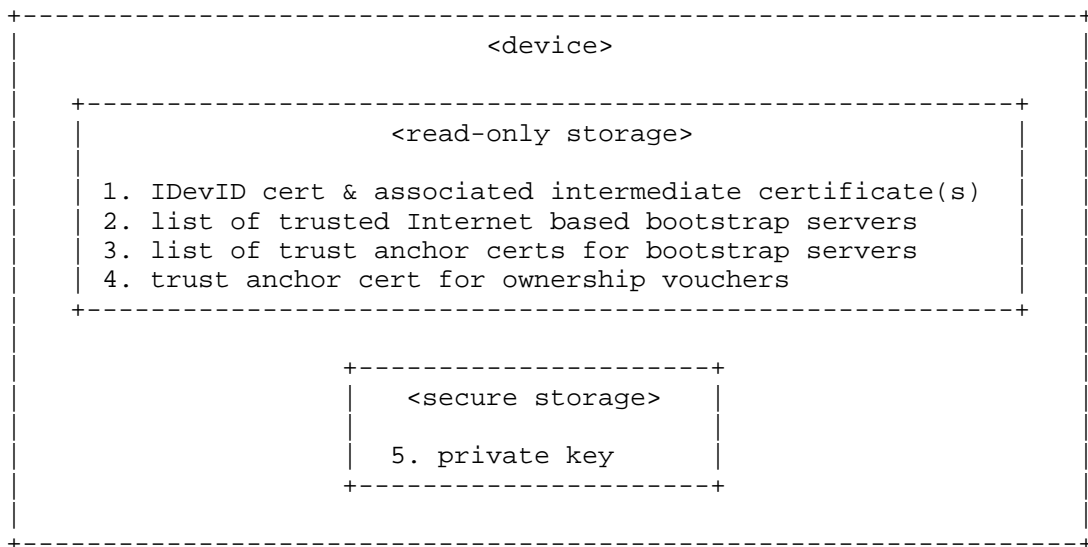


5. After having tried all supported sources of bootstrapping data, the device MAY retry again all the sources and/or provide manageability interfaces for manual configuration (e.g., CLI, HTTP, NETCONF, etc.). If manual configuration is allowed, and such configuration is provided, the device MUST immediately cease trying to obtain bootstrapping data, as it would then no longer be in running its factory default configuration.

## 8. Device Details

Devices supporting the bootstrapping strategy described in this document MUST have the preconfigured factory default state and bootstrapping logic described in the following sections.

### 8.1. Factory Default State



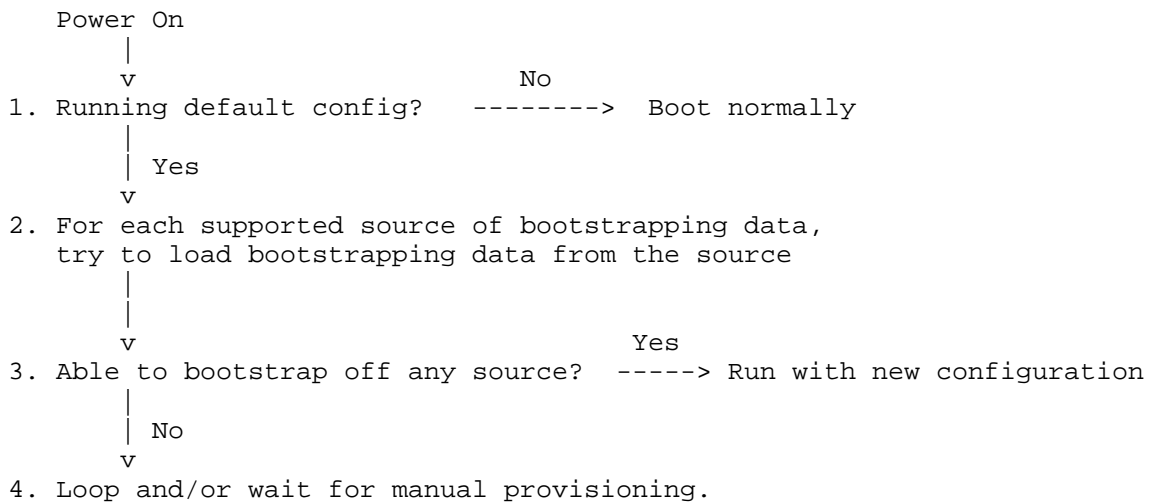
Each numbered item below corresponds to a numbered item in the diagram above.

1. Devices MUST be manufactured with an initial device identifier (IDevID), as defined in [Std-802.1AR-2009]. The IDevID is an X.509 certificate, encoding the device's unique device identifier (e.g., serial number). The device MUST also possess any intermediate certificates between the IDevID certificate and the manufacturer's IDevID trust anchor certificate, which is provided to prospective owners separately (e.g., Section 7.1).

2. Devices that support loading bootstrapping data from an Internet-based bootstrap server (see Section 6.4) MUST be manufactured with a configured list of trusted bootstrap servers. Consistent with redirect information (Section 3.1, each bootstrap server MAY be identified by its hostname or IP address, and an optional port.
3. Devices that support loading bootstrapping data from an Internet-based bootstrap server (see Section 6.4) MUST also be manufactured with a list of trust anchor certificates that can be used for X.509 certificate path validation ([RFC6125], Section 6) on the bootstrap server's TLS server certificate.
4. Devices that support loading owner signed data (see Section 1.2) MUST also be manufactured with the trust anchor certificate for the ownership vouchers.
5. Device MUST be manufactured with a private key that corresponds to the public key encoded in the device's IDevID certificate. This private key SHOULD be securely stored, ideally by a cryptographic processor (e.g., a TPM).

## 8.2. Boot Sequence

A device claiming to support the bootstrapping strategy defined in this document MUST support the boot sequence described in this section.



Each numbered item below corresponds to a numbered item in the diagram above.

1. When the device powers on, it first checks to see if it is running the factory default configuration. If it is running a modified configuration, then it boots normally.
2. The device iterates over its list of sources for bootstrapping data (Section 6). Details for how to process a source of bootstrapping data are provided in Section 8.3.
3. If the device is able to bootstrap itself off any of the sources of bootstrapping data, it runs with the new bootstrapped configuration.
4. Otherwise the device MAY loop back through the list of bootstrapping sources again and/or wait for manual provisioning.

### 8.3. Processing a Source of Bootstrapping Data

This section describes a recursive algorithm that a device claiming to support the bootstrapping strategy defined in this document MUST use to authenticate bootstrapping data. A device enters this algorithm for each new source of bootstrapping data. The first time the device enters this algorithm, it MUST initialize a conceptual trust state variable, herein referred to as "trust-state", to FALSE. The ultimate goal of this algorithm is for the device to process bootstrap information (Section 3.2) while the trust-state variable is TRUE.

If the data source is a bootstrap server, and the device is able to authenticate the server using X.509 certificate path validation ([RFC6125], Section 6) to one of the device's preconfigured trust anchors, or to a trust anchor that it learned from a previous step, then the device MUST set trust-state to TRUE.

If trust-state is TRUE, when connecting to the bootstrap server, the device MUST use its IDevID certificate for client certificate based authentication and MUST POST progress notifications using the bootstrap server's "notification" action. Otherwise, if trust-state is FALSE, when connecting to the bootstrap server, the device MUST NOT use its IDevID certificate for a client certificate based authentication and MUST NOT POST progress notifications using the bootstrap server's "notification" action.

When accessing a bootstrap server, the device MUST only access its top-level resource, to obtain all the data staged for it in one GET request, so that it can determine if the data is signed or not, and

thus act accordingly. If trust-state is TRUE, then the device MAY also access the bootstrap servers 'notification' resource for the device.

For any source of bootstrapping data (e.g., Section 6), if the data is signed and the device is able to validate the signed data using the algorithm described in Section 8.4, then the device MUST set trust-state to TRUE, else the device MUST set trust-state to FALSE. Note, this is worded to cover the special case when signed data is returned even from a trusted bootstrap server.

If the data is bootstrap information (not redirect information), and trust-state is FALSE, the device MUST exit the recursive algorithm, returning to the state machine described in Section 8.2. Otherwise, the device MUST attempt to process the bootstrap information as described in Section 8.6. In either case, success or failure, the device MUST exit the recursive algorithm, returning to the state machine described in Section 8.2, the only difference being in how it responds to the "Able to bootstrap off any source?" conditional described in that state machine.

If the data is redirect information, the device MUST process the redirect information as described in Section 8.5. This is the recursion step, it will cause the device to reenter this algorithm, but this time the data source will most definitely be a bootstrap server, as that is all redirect information is able to do.

#### 8.4. Validating Signed Data

Whenever a device is presented signed data from an untrusted source, it MUST validate the signed data as described in this section. If the signed data is provided by a trusted source, a redundant trust case, the device MAY skip verifying the signature.

Whenever there is signed data, the device MUST also be provided an ownership voucher and an owner certificate. Depending on circumstances, the device MAY also be provided certificate revocations. How all the needed artifacts are provided for each source of bootstrapping data is defined in Section 6.

The device MUST first authenticate the ownership voucher by validating the signature on it to one of its preconfigured trust anchors (see Section 8.1) and verify that the voucher contains the device's unique identifier (e.g., serial number). If the voucher contains an expiration timestamp, the device MUST also verify that the voucher has not expired. If the authentication of the voucher is successful, the device extracts from it information that can be used to verify the owner certificate in the next step.

Next the device MUST authenticate the owner certificate by performing X.509 certificate path verification to the trusted certificate provided in the voucher. If the device insists on verifying revocation status, it MUST also verify that none of the certificates in the chain of certificates have been revoked or expired. If the authentication of the certificate is successful, the device extracts the owner's public key from the certificate for use in the next step.

Finally the device MUST verify the signature over information artifact was generated by the private key matching the public key extracted from the owner certificate in the previous step.

If any of these steps fail, then the device MUST mark the data as invalid and not perform any of the subsequent steps.

#### 8.5. Processing Redirect Information

In order to process redirect information (Section 3.1), the device MUST follow the steps presented in this section.

Processing redirect information is straightforward. The device sequentially steps through the list of provided bootstrap servers until it can find one it can bootstrap off of.

If a hostname is provided, and the hostname's DNS resolution is to more than one IP address, the device MUST attempt to connect to all of the DNS resolved addresses at least once, before moving on to the next bootstrap server. If the device is able to obtain bootstrapping data from any of the DNS resolved addresses, it MUST immediately process that data, without attempting to connect to any of the other DNS resolved addresses.

If the redirect information is trusted (e.g., trust-state is TRUE), and the bootstrap server entry contains a trust anchor certificate, then the device MUST authenticate the bootstrap server using X.509 certificate path validation ([RFC6125], Section 6) to the specified trust anchor. If the device is unable to authenticate the bootstrap server to the specified trust anchor, the device MUST NOT attempt a provisional connection to the bootstrap server (i.e., by blindly accepting its server certificate).

If the redirect information is untrusted (e.g., trust-state is FALSE), the device MUST discard any trust anchors provided by the redirect information and establish a provisional connection to the bootstrap server (i.e., by blindly accepting its TLS server certificate).

## 8.6. Processing Bootstrap Information

In order to process bootstrap information (Section 3.2), the device MUST follow the steps presented in this section.

When processing bootstrap information, the device MUST first process the boot image information, then execute the pre-configuration script (if any), then commit the initial configuration, and then execute the script (if any), in that order. If the device encounters an error at any step, it MUST NOT proceed to the next step.

First the device MUST determine if the image it is running satisfies the specified boot image criteria (e.g., name or fingerprint match). If it does not, the device MUST download, verify, and install the specified boot image, and then reboot. To verify the boot image, the device MUST check that the boot image file matches the fingerprint (e.g., sha256) supplied by the bootstrapping information. Upon rebooting, the device MUST still be in its factory default state, causing the bootstrapping process to run again, which will eventually come to this very point, but this time the device's running image will satisfy the specified criteria, and thus the device will move to processing the next step.

Next, for devices that support executing scripts, if a pre-configuration script has been specified, the device MUST execute the script and check its exit status code to determine if had any warnings or errors. In the case of errors, the device MUST reset itself in such a way that force the reinstallation of its boot image, thereby wiping out any bad state the script might have left behind.

Next the device commits the provided initial configuration. Assuming no errors, the device moves to processing the next step.

Again, for devices that support executing scripts, if a post-configuration script has been specified, the device MUST execute the script and check its exit status code to determine if it had any warnings or errors. In the case of errors, the device MUST reset itself in such a way that force the reinstallation of its boot image, thereby wiping out any bad state the script might have left behind.

At this point, the device has completely processed the bootstrapping data and is ready to be managed. If the device obtained the bootstrap information from a trusted bootstrap server, the device MUST send the 'bootstrap-complete' notification now.

At this point the device is configured and no longer running its factory default configuration. Notably, if the bootstrap information

configured the device it initiate a call home connection, the device would proceed to do so now.

## 9. The Zero Touch Information Artifact

This section defines a YANG [RFC6020] module that is used to define the data model for the Zero Touch Information artifact described in Section 4.1. Examples illustrating this artifact in use are provided in Section 9.2.

### 9.1. Tree Diagram

The following tree diagram provides an overview of the data model for the Zero Touch Information artifact. The syntax used for this tree diagram is described in Section 1.4.

```

module: ietf-zerotouch-information
+---- (information-type)
+---:(redirect-information)
|   +---- redirect-information
|   |   +---- bootstrap-server* [address]
|   |   |   +---- address          inet:host
|   |   |   +---- port?           inet:port-number
|   |   |   +---- trust-anchor?   binary
|   +---:(bootstrap-information)
|   |   +---- bootstrap-information
|   |   |   +---- boot-image
|   |   |   |   +---- name          string
|   |   |   |   +---- (hash-algorithm)
|   |   |   |   |   +---:(sha256)
|   |   |   |   |   |   +---- sha256?  string
|   |   |   |   +---- uri*         inet:uri
|   |   +---- configuration-handling? enumeration
|   |   +---- pre-configuration-script? script
|   |   +---- configuration?
|   |   +---- post-configuration-script? script

```

### 9.2. Example Usage

This section presents examples for how the information artifact (Section 4.1) can be encoded into a document that can be distributed outside the bootstrap server's RESTCONF API.

The following example illustrates how redirect information can be encoded into an artifact.

```

<redirect-information
  xmlns="urn:ietf:params:xml:ns:yang:ietf-zerotouch-information">
  <bootstrap-server>
    <address>phs1.example.com</address>
    <port>8443</port>
    <trust-anchor>Base64-encoded X.50 </trust-anchor>
  </bootstrap-server>
  <bootstrap-server>
    <address>phs2.example.com</address>
    <port>8443</port>
    <trust-anchor>Base64-encoded X.50 </trust-anchor>
  </bootstrap-server>
  <bootstrap-server>
    <address>phs3.example.com</address>
    <port>8443</port>
    <trust-anchor>Base64-encoded X.50 </trust-anchor>
  </bootstrap-server>
</redirect-information>

```

The following example illustrates how bootstrap information can be encoded into an artifact. This example uses datamodels from [RFC7317] and [I-D.ietf-netconf-netconf-client-server].

<-- '\ ' line wrapping added for formatting purposes only -->

```

<bootstrap-information
  xmlns="urn:ietf:params:xml:ns:yang:ietf-zerotouch-information">
  <boot-image>
    <name>boot-image-v3.2R1.6.img</name>
    <sha256>Hex-encoded SHA256 hash</sha256>
    <uri>file:///some/path/to/raw/file </uri>
  </boot-image>
  <configuration-handling>merge</configuration-handling>
  <configuration>
    <!-- from ietf-system.yang -->
    <system xmlns="urn:ietf:params:xml:ns:yang:ietf-system">
      <authentication>
        <user>
          <name>admin</name>
          <authorized-key>
            <name>admin's rsa ssh host-key</name>
            <algorithm>ssh-rsa</algorithm>
            <key-data>AAAAB3NzaC1yc2EAAAADAQABAAQDeJMV8zrtsi8CgEsRC\
jCzfve2m6zD3awSBPrh7ICgglQvHVbPL89eHLuecStKL3HrEgXaI/O2Mwj\
E1lG9YxLzeS5p2ngzK6lvikUSqfMukeBohFTrDZ8bUtrF+HMLlTRnoCVcC\
WAw1lOr9IDGAuww6G45gLCHalHMmBtQxKnZdzU9kx/fL3ZS5G76Fy6sA5\
vg7SLqQFPjXXft2CAhin8xwYRZy6r/2N9PMJ2Dnepvq4H2DKqBIe340jWq\

```



```

        EIuA7LvEJYql4unq4Iog+/+CiumTkmQIWRgIoJ4FCzYkO9NvRE6fOSLLf6\
        gakWVOZZgQ8929uWjCWlGlqn2mPibp2Go1</key-data>
    </authorized-key>
</user>
</authentication>
</system>
<!-- from ietf-netconf-server.yang -->
<netconf-server
  xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-server">
  <call-home>
    <netconf-client>
      <name>config-mgr</name>
      <ssh>
        <endpoints>
          <endpoint>
            <name>east-data-center</name>
            <address>11.22.33.44</address>
          </endpoint>
          <endpoint>
            <name>west-data-center</name>
            <address>55.66.77.88</address>
          </endpoint>
        </endpoints>
        <host-keys>
          <host-key>
            <name>certificate</name>
            <certificate>builtin-idevid-cert</certificate>
          </host-key>
        </host-keys>
        <client-cert-auth>
          <trusted-ca-certs>deployment-specific-ca-certs</trusted-ca-certs>
          <trusted-client-certs>explicitly-trusted-client-certs</trusted-clie
ent-certs>
        </client-cert-auth>
      </ssh>
    <connection-type>
      <periodic>
        <idle-timeout>300</idle-timeout>
        <reconnect-timeout>60</reconnect-timeout>
      </periodic>
    </connection-type>
    <reconnect-strategy>
      <start-with>last-connected</start-with>
      <max-attempts>3</max-attempts>
    </reconnect-strategy>
  </netconf-client>
</call-home>
</netconf-server>
</configuration>

```

</bootstrap-information>

### 9.3. YANG Module

The Zero Touch Information artifact is normatively defined by the YANG module defined in this section.

Note: the module defined herein uses data types defined in [RFC5280], [RFC6234], and [RFC6991].

<CODE BEGINS> file "ietf-zerotouch-information@2017-03-13.yang"

```
module ietf-zerotouch-information {
  yang-version "1.1";

  namespace "urn:ietf:params:xml:ns:yang:ietf-zerotouch-information";
  prefix    "zti";

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

  import ietf-restconf {
    prefix rc;
    description
      "This import statement is only present to access
       the yang-data extension defined in RFC 8040.";
    reference "RFC 8040: RESTCONF Protocol";
  }

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

  contact
    "WG Web:  http://tools.ietf.org/wg/netconf
     WG List:  <mailto:netconf@ietf.org>
     Author:   Kent Watsen <mailto:kwatsen@juniper.net>";

  description
    "This module defines the data model for the Zero Touch Information
     artifact defined by RFC XXXX: Zero Touch Provisioning for NETCONF
     or RESTCONF based Management.

     Copyright (c) 2014 IETF Trust and the persons identified as
     authors of the code. All rights reserved.

     Redistribution and use in source and binary forms, with or without
```

modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>).

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

```
revision "2017-03-13" {
  description
    "Initial version";
  reference
    "RFC XXXX: Zero Touch Provisioning for NETCONF or RESTCONF based
    Management";
}

rc:yang-data zerotouch-information {
  uses zerotouch-information-grouping;
}

grouping zerotouch-information-grouping {
  description
    "Defines the zerotouch information data model. Grouping
    exists only to enable pyang tree output.";

  choice information-type {
    mandatory true;
    description
      "This choice statement ensures the response only contains
      redirect-information or bootstrap-information. Note that
      this is the only mandatory true node, as the other nodes
      are not needed when the device trusts the bootstrap server,
      in which case the data does not need to be signed.";

    container redirect-information {
      description
        "This is redirect information, as described in Section 3.1
        in RFC XXXX. Its purpose is to redirect a device to another
        bootstrap server.";
      reference
        "RFC XXXX: Zero Touch Provisioning for NETCONF or RESTCONF
        based Management";

      list bootstrap-server {
        key address;
        description
          "A bootstrap server entry.";
      }
    }
  }
}
```

```
    leaf address {
      type inet:host;
      mandatory true;
      description
        "The IP address or hostname of the bootstrap server the
        device should redirect to.";
    }
    leaf port {
      type inet:port-number;
      default 443;
      description
        "The port number the bootstrap server listens on.";
    }
    leaf trust-anchor { //should there be two fields like voucher?
      type binary;
      description
        "An X.509 v3 certificate structure as specified by RFC
        5280, Section 4, encoded using ASN.1 distinguished
        encoding rules (DER), as specified in ITU-T X.690. A
        certificate that a device can use as a trust anchor
        to authenticate the bootstrap server it is being
        redirected to.";
      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).";
    }
  }
}

container bootstrap-information {

  description
    "This is bootstrap information, as described in Section 3.2 in
    RFC XXXX. Its purpose is to provide the device everything it
    needs to bootstrap itself.";
  reference
    "RFC XXXX: Zero Touch Provisioning for NETCONF or RESTCONF
    based Management";

  container boot-image {
    description
      "Specifies criteria for the boot image the device MUST
```

```
        be running.";

    leaf name { // maybe this should be a regex?
        type string;
        mandatory true;
        description
            "The name of a software image that the device MUST
             be running in order to process the remaining nodes.";
    }
    choice hash-algorithm {
        mandatory true;
        description
            "Identifies the hash algorithm used.";
        leaf sha256 {
            type string;
            description
                "The hex-encoded SHA-256 hash over the boot
                 image file. This is used by the device to
                 verify a downloaded boot image file.";
            reference
                "RFC 6234: US Secure Hash Algorithms.";
        }
    }
    leaf-list uri {
        type inet:uri;
        min-elements 1;
        description
            "An ordered list of URIs to where the boot-image file MAY
             be obtained. Deployments MUST know in which URI schemes
             (http, ftp, etc.) a device supports. If a secure scheme
             (e.g., https) is provided, a device MAY establish a
             provisional connection to the server, by blindly
             accepting the server's credentials (e.g., its TLS
             certificate)";
    }
}

leaf configuration-handling {
    type enumeration {
        enum merge {
            description
                "Merge configuration into existing running configuration.";
        }
        enum replace {
            description
                "Replace existing running configuration with the passed
                 configuration.";
        }
    }
}
```

```
    }
    description
        "This enumeration indicates how the server should process
        the provided configuration.  When not specified, the device
        MAY determine how to process the configuration using other
        means (e.g., vendor-specific metadata).";
    }

    leaf pre-configuration-script {
        type script;
        description
            "A script that, when present, is executed before the
            configuration has been processed.";
    }

    anydata configuration {
        must "../configuration-handling";
        description
            "Any configuration data model known to the device.  It may
            contain manufacturer-specific and/or standards-based data
            models.";
    }

    leaf post-configuration-script {
        type script;
        description
            "A script that, when present, is executed after the
            configuration has been processed.";
    }
}
}
```

```
typedef script {
    type binary;
    description
        "A device specific script that enables the execution of commands
        to perform actions not possible thru configuration alone.

        No attempt is made to standardize the contents, running context,
        or programming language of the script.  The contents of the
        script are considered specific to the vendor, product line,
        and/or model of the device.

        If a script is erroneously provided to a device that does not
        support the execution of scripts, the device SHOULD send a
        'script-warning' notification message, but otherwise continue
        processing the bootstrapping data as if the script had not
```

been present.

The script returns exit status code '0' on success and non-zero on error, with accompanying stderr/stdout for logging purposes. In the case of an error, the exit status code will specify what the device should do.

If the exit status code is greater than zero, then the device should assume that the script had a soft error, which the script believes does not affect manageability. If the device obtained the bootstrap information from a bootstrap server, it SHOULD send a 'script-warning' notification message.

If the exit status code is less than zero, the device should assume the script had a hard error, which the script believes will affect manageability. In this case, the device SHOULD send a 'script-error' notification message followed by a reset that will force a new boot-image install (wiping out anything the script may have done) and restart the entire bootstrapping process again.";

}

}

<CODE ENDS>

## 10. The Zero Touch Bootstrap Server API

This section defines a YANG [RFC6020] module that is used to define the RESTCONF [RFC8040] API used by the bootstrap server defined in Section 6.4. Examples illustrating this API in use are provided in Section 10.2.

### 10.1. Tree Diagram

The following tree diagram provides an overview for the bootstrap server RESTCONF API. The syntax used for this tree diagram is described in Section 1.4.

```

module: ietf-zerotouch-bootstrap-server
  +--ro device* [unique-id]
    +--ro unique-id          string
    +--ro zerotouch-information pkcs7
    +--ro owner-certificate?  pkcs7
    +--ro ownership-voucher?  pkcs7
    +---x notification
      +---w input
        +---w notification-type enumeration
        +---w message?         string
        +---w ssh-host-keys
          +---w ssh-host-key*
            +---w format        enumeration
            +---w key-data      string
        +---w trust-anchors
          +---w trust-anchor*
            +---w protocol*     enumeration
            +---w certificate   pkcs7

```

In the above diagram, notice that all of the protocol accessible nodes are read-only, to assert that devices can only pull data from the bootstrap server.

Also notice that the module defines an action statement, which devices use to provide progress notifications to the bootstrap server.

## 10.2. Example Usage

This section presents some examples illustrating the bootstrap server's API. Two examples are provided, one illustrating a device fetching bootstrapping data from the server, and the other illustrating a data posting a progress notification to the server.

The following example illustrates a device using the API to fetch its bootstrapping data from the bootstrap server. In this example, the device receives a signed response; an unsigned response would look similar except the last two fields (owner-certificate and ownership-voucher) would be absent in the response.



## REQUEST

-----

['\ ' line wrapping added for formatting only]

```
GET https://example.com/restconf/data/ietf-zero-touch-bootstrap-server:\
device=123456 HTTP/1.1
HOST: example.com
Accept: application/yang.data+xml
```

## RESPONSE

-----

```
HTTP/1.1 200 OK
Date: Sat, 31 Oct 2015 17:02:40 GMT
Server: example-server
Content-Type: application/yang.data+xml
```

&lt;!-- '\ ' line wrapping added for formatting purposes only --&gt;

```
<device
  xmlns="urn:ietf:params:xml:ns:yang:ietf-zero-touch-bootstrap-server">
  <unique-id>123456789</unique-id>
  <zerotouch-information>\
    Base64-encoded Zero Touch Information artifact\
  </zerotouch-information>
  <owner-certificate>Base64-encoded PKCS#7</owner-certificate>
  <ownership-voucher>Base64-encoded Voucher</ownership-voucher>
</device>
```

The following example illustrates a device using the API to post a notification to a bootstrap server. Illustrated below is the 'bootstrap-complete' message, but the device may send other notifications to the server while bootstrapping (e.g., to provide status updates). In this message, the device is sending both its SSH host keys and TLS server certificate, which the bootstrap server may, for example, pass to an NMS, as discussed in Section 7.3.

Note that devices that are able to present an IDevID certificate [Std-802.1AR-2009] when establishing SSH or TLS connections do not need to include its DevID certificate in the bootstrap-complete message. It is unnecessary to send the DevID certificate in this case because the IDevID certificate does not need to be pinned by an NMS in order to be trusted.

Note that the bootstrap server MUST NOT process a notification from a device without first authenticating the device. This is in contrast to when a device is fetching data from the server, a read-only

operation, in which case device authentication is not strictly required (e.g., when sending signed information).

## REQUEST

-----

['\' line wrapping added for formatting only]

POST https://example.com/restconf/data/ietf-zero-touch:\  
device=123456/notification HTTP/1.1  
HOST: example.com  
Content-Type: application/yang.data+xml

<!-- \' line wrapping added for formatting purposes only -->

```
<input
  xmlns="urn:ietf:params:xml:ns:yang:ietf-zero-touch-bootstrap-server">
  <notification-type>bootstrap-complete</notification-type>
  <message>example message</message>
  <ssh-host-keys>
    <ssh-host-key>
      <format>ssh-rsa</format>
      <key-data>Base64-encoded SSH RSA Public Key</key-data>
    </ssh-host-key>
    <ssh-host-key>
      <format>ssh-dsa</format>
      <key-data>Base64-encoded SSH DSA Public Key</key-data>
    </ssh-host-key>
  </ssh-host-keys>
  <trust-anchors>
    <trust-anchor>
      <protocol>netconf-ssh</protocol>
      <protocol>netconf-tls</protocol>
      <protocol>restconf-tls</protocol>
      <protocol>netconf-ch-ssh</protocol>
      <protocol>netconf-ch-tls</protocol>
      <protocol>restconf-ch-tls</protocol>
      <certificate>Base64-encoded X.509</certificate>
    </trust-anchor>
  </trust-anchors>
</input>
```

## RESPONSE

-----

HTTP/1.1 204 No Content  
Date: Sat, 31 Oct 2015 17:02:40 GMT  
Server: example-server

### 10.3. YANG Module

The bootstrap server's device-facing API is normatively defined by the YANG module defined in this section.

Note: the module defined herein uses data types defined in [RFC2315], [RFC5280], and [I-D.ietf-anima-voucher].

<CODE BEGINS> file "ietf-zerotouch-bootstrap-server@2017-03-13.yang"

```
module ietf-zerotouch-bootstrap-server {
  yang-version "1.1";

  namespace
    "urn:ietf:params:xml:ns:yang:ietf-zerotouch-bootstrap-server";
  prefix      "ztbs";

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

  contact
    "WG Web:      <http://tools.ietf.org/wg/netconf/>
    WG List:      <mailto:netconf@ietf.org>
    Author:       Kent Watsen
                  <mailto:kwatsen@juniper.net>";

  description
    "This module defines an interface for bootstrap servers, as defined
    by RFC XXXX: Zero Touch Provisioning for NETCONF or RESTCONF based
    Management.

    Copyright (c) 2014 IETF Trust and the persons identified as
    authors of the code. All rights reserved.

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

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

  revision "2017-03-13" {
    description
      "Initial version";
    reference
      "RFC XXXX: Zero Touch Provisioning for NETCONF or RESTCONF based
```

```
    Management";
}

// typedefs

typedef pkcs7 {
    type binary;
    description
        "A PKCS #7 SignedData structure, as specified by Section 9.1
        in RFC 2315, encoded using ASN.1 distinguished encoding rules
        (DER), as specified in ITU-T X.690.";
    reference
        "RFC 2315:
        PKCS #7: Cryptographic Message Syntax Version 1.5.
        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).";
}

// protocol accessible node

list device {
    key unique-id;
    config false;

    description
        "A device's record entry. This is the only RESTCONF resource
        that a device will GET, as described in Section 8.2 in RFC XXXX.
        Getting just this top-level node provides a device with all the
        data it needs in a single request.";
    reference
        "RFC XXXX: Zero Touch Provisioning for NETCONF or
        RESTCONF based Management";

    leaf unique-id {
        type string;
        description
            "A unique identifier for the device (e.g., serial number).
            Each device accesses its bootstrapping record by its unique
            identifier.";
    }

    leaf zerotouch-information {
        type pkcs7;
        mandatory true;
        description
```

```
"A 'zerotouch-information' artifact, as described in Section
4.1 of RFC XXXX. When conveyed over an untrusted transport, in
order to be processed by a device, this PKCS#7 SignedData
structure MUST contain a 'signerInfo' structure, described
in Section 9.1 of RFC 2315, containing a signature generated
using the owner's private key.";
reference
  "RFC XXXX: Zero Touch Provisioning for NETCONF or
  RESTCONF based Management.
  RFC 2315:
    PKCS #7: Cryptographic Message Syntax Version 1.5";
}

leaf owner-certificate {
  type pkcs7;
  must "../zerotouch-information" {
    description
      "An 'zerotouch-information' artifact must be present
      whenever an ownership certificate is presented.";
  }
  description
    "An unsigned PKCS #7 SignedData structure, as specified by
    Section 9.1 in RFC 2315, encoded using ASN.1 distinguished
    encoding rules (DER), as specified in ITU-T X.690.

    This structure MUST contain the owner certificate and all
    intermediate certificates leading up to at least the trust
    anchor certificate specified in the ownership voucher.
    Additionally, if needed by the device, this structure MAY
    also contain suitably fresh CRL and or OCSP Responses.

    X.509 certificates and CRLs are described in RFC 5280.
    OCSP Responses are described in RFC 6960.";
  reference
    "RFC 2315:
      PKCS #7: Cryptographic Message Syntax Version 1.5.
    RFC 5280:
      Internet X.509 Public Key Infrastructure Certificate
      and Certificate Revocation List (CRL) Profile.
    RFC 6960:
      X.509 Internet Public Key Infrastructure Online
      Certificate Status Protocol - OCSP.
    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 ownership-voucher {
  type pkcs7;
  must "../owner-certificate" {
    description
      "An owner certificate must be present whenever an ownership
       voucher is presented.";
  }
  description
    "A 'voucher' artifact, as described in Section 5 of
     I-D.ietf-anima-voucher. The voucher's 'device-identifier'
     MUST reference both the device's unique identifier and
     the owner's owner certificate.";
  reference
    "I-D.ietf-anima-voucher:
     Voucher and Voucher Revocation Profiles for Bootstrapping
     Protocols";
}

action notification {
  input {
    leaf notification-type {
      type enumeration {
        enum bootstrap-initiated {
          description
            "Indicates that the device has just accessed the
             bootstrap server. The 'message' field below MAY
             contain any additional information that the
             manufacturer thinks might be useful.";
        }
        enum parsing-warning {
          description
            "Indicates that the device had a non-fatal error when
             parsing the response from the bootstrap server. The
             'message' field below SHOULD indicate the specific
             warning that occurred.";
        }
        enum parsing-error {
          description
            "Indicates that the device encountered a fatal error
             when parsing the response from the bootstrap server.
             For instance, this could be due to malformed encoding,
             the device expecting signed data when only unsigned
             data is provided, because the ownership voucher didn't
             include the device's unique identifier, or because the
             signature didn't match. The 'message' field below
             SHOULD indicate the specific error. This notification
             also indicates that the device has abandoned trying to
             bootstrap off this bootstrap server.";
        }
      }
    }
  }
}
```

```
}
enum boot-image-warning {
  description
    "Indicates that the device encountered a non-fatal
    error condition when trying to install a boot-image.
    A possible reason might include a need to reformat a
    partition causing loss of data. The 'message' field
    below SHOULD indicate any warning messages that were
    generated.";
}
enum boot-image-error {
  description
    "Indicates that the device encountered an error when
    trying to install a boot-image, which could be for
    reasons such as a file server being unreachable,
    file not found, signature mismatch, etc. The
    'message' field SHOULD indicate the specific error
    that occurred. This notification also indicates
    that the device has abandoned trying to bootstrap
    off this bootstrap server.";
}
enum pre-script-warning {
  description
    "Indicates that the device obtained a greater than
    zero exit status code from the script when it was
    executed. The 'message' field below SHOULD indicate
    both the resulting exit status code, as well as
    capture any stdout/stderr messages the script may
    have produced.";
}
enum pre-script-error {
  description
    "Indicates that the device obtained a less than zero
    exit status code from the script when it was executed.
    The 'message' field below SHOULD indicate both the
    resulting exit status code, as well as capture any
    stdout/stderr messages the script may have produced.
    This notification also indicates that the device has
    abandoned trying to bootstrap off this bootstrap
    server.";
}
enum config-warning {
  description
    "Indicates that the device obtained warning messages
    when it committed the initial configuration. The
    'message' field below SHOULD indicate any warning
    messages that were generated.";
}
```

```
enum config-error {
  description
    "Indicates that the device obtained error messages
    when it committed the initial configuration. The
    'message' field below SHOULD indicate the error
    messages that were generated. This notification
    also indicates that the device has abandoned trying
    to bootstrap off this bootstrap server.";
}
enum post-script-warning {
  description
    "Indicates that the device obtained a greater than
    zero exit status code from the script when it was
    executed. The 'message' field below SHOULD indicate
    both the resulting exit status code, as well as
    capture any stdout/stderr messages the script may
    have produced.";
}
enum post-script-error {
  description
    "Indicates that the device obtained a less than zero
    exit status code from the script when it was executed.
    The 'message' field below SHOULD indicate both the
    resulting exit status code, as well as capture any
    stdout/stderr messages the script may have produced.
    This notification also indicates that the device has
    abandoned trying to bootstrap off this bootstrap
    server.";
}
enum bootstrap-complete {
  description
    "Indicates that the device successfully processed the
    all the bootstrapping data and that it is ready to be
    managed. The 'message' field below MAY contain any
    additional information that the manufacturer thinks
    might be useful. After sending this notification,
    the device is not expected to access the bootstrap
    server again.";
}
enum informational {
  description
    "Indicates any additional information not captured by
    any of the other notification-type. For instance, a
    message indicating that the device is about to reboot
    after having installed a boot-image could be provided.
    The 'message' field below SHOULD contain information
    that the manufacturer thinks might be useful.";
}
```



```
    }
    mandatory true;
    description
        "The type of notification provided.";
}
leaf message {
    type string;
    description
        "An optional human-readable value.";
}
container ssh-host-keys {
    when "../notification-type = 'bootstrap-complete'" {
        description
            "SSH host keys are only sent when the notification
            type is 'bootstrap-complete'.";
    }
    description
        "A list of SSH host keys an NMS may use to authenticate
        a NETCONF connection to the device with.";
    list ssh-host-key {
        description
            "An SSH host-key";
        leaf format {
            type enumeration {
                enum ssh-dss { description "ssh-dss"; }
                enum ssh-rsa { description "ssh-rsa"; }
            }
            mandatory true;
            description
                "The format of the SSH host key.";
        }
        leaf key-data {
            type string;
            mandatory true;
            description
                "The key data for the SSH host key";
        }
    }
}
container trust-anchors {
    when "../notification-type = 'bootstrap-complete'" {
        description
            "Trust anchors are only sent when the notification
            type is 'bootstrap-complete'.";
    }
    description
        "A list of trust anchor certificates an NMS may use to
        authenticate a NETCONF or RESTCONF connection to the
```

```
    device with.";
list trust-anchor {
  description
    "A list of trust anchor certificates an NMS may use to
    authenticate a NETCONF or RESTCONF connection to the
    device with.";
  leaf-list protocol {
    type enumeration {
      enum netconf-ssh      { description "netconf-ssh"; }
      enum netconf-tls      { description "netconf-tls"; }
      enum restconf-tls     { description "restconf-tls"; }
      enum netconf-ch-ssh   { description "netconf-ch-ssh"; }
      enum netconf-ch-tls   { description "netconf-ch-tls"; }
      enum restconf-ch-tls  { description "restconf-ch-tls"; }
    }
    min-elements 1;
    description
      "The protocols that this trust anchor secures.";
  }
  leaf certificate {
    type pkcs7;
    mandatory true;
    description
      "An X.509 v3 certificate structure, as specified by
      Section 4 in RFC5280, 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).";
  }
}
} // end action
} // end device
}
```

<CODE ENDS>

## 11. Security Considerations

### 11.1. Immutable storage for trust anchors

Devices **MUST** ensure that all their trust anchor certificates, including those for connecting to bootstrap servers and verifying ownership vouchers, are protected from external modification.

It may be necessary to update these certificates over time (e.g., the manufacturer wants to delegate trust to a new CA). It is therefore expected that devices **MAY** update these trust anchors when needed through a verifiable process, such as a software upgrade using signed software images.

### 11.2. Clock Sensitivity

The solution in this document relies on TLS certificates, owner certificates, and ownership vouchers, all of which require an accurate clock in order to be processed correctly (e.g., to test validity dates and revocation status). Implementations **MUST** ensure devices have an accurate clock when shipped from manufacturing facilities, and take steps to prevent clock tampering.

If it is not possible to ensure clock accuracy, it is **RECOMMENDED** that implementations disable the aspects of the solution having clock sensitivity. In particular, such implementations should assume that TLS certificates and owner certificates are not revokable. In real-world terms, this means that manufacturers **SHOULD** only issue a single ownership voucher for the lifetime of some devices.

It is important to note that implementations **SHOULD NOT** rely on NTP for time, as it is not a secure protocol.

### 11.3. Blindly authenticating a bootstrap server

This document allows a device to blindly authenticate a bootstrap server's TLS certificate. It does so to allow for cases where the redirect information may be obtained in an unsecured manner, which is desirable to support in some cases.

To compensate for this, this document requires that devices, when connected to an untrusted bootstrap server, do not send their IDevID certificate for client authentication, and they do not POST any progress notifications, and they assert that data downloaded from the server is signed.

#### 11.4. Entropy loss over time

Section 7.2.7.2 of the IEEE Std 802.1AR-2009 standard says that IDevID certificate should never expire (i.e. having the notAfter value 99991231235959Z). Given the long-lived nature of these certificates, it is paramount to use a strong key length (e.g., 512-bit ECC).

#### 11.5. Serial Numbers

This draft suggests using the device's serial number as the unique identifier in its IDevID certificate. This is because serial numbers are ubiquitous and prominently contained in invoices and on labels affixed to devices and their packaging. That said, serial numbers many times encode revealing information, such as the device's model number, manufacture date, and/or sequence number. Knowledge of this information may provide an adversary with details needed to launch an attack.

#### 11.6. Sequencing Sources of Bootstrapping Data

For devices supporting more than one source for bootstrapping data, no particular sequencing order has to be observed for security reasons, as the solution for each source is considered equally secure. However, from a privacy perspective, it is RECOMMENDED that devices access local sources before accessing remote sources.

### 12. IANA Considerations

#### 12.1. The BOOTP Manufacturer Extensions and DHCP Options Registry

The following registrations are in accordance to RFC 2939 [RFC2939] for "BOOTP Manufacturer Extensions and DHCP Options" registry maintained at <http://www.iana.org/assignments/bootp-dhcp-parameters>.

Tag: XXX

Name: Zero Touch Information

Returns up to three zero touch bootstrapping artifacts.

Code	Len
XXX	n
zerotouch-information	
owner-certificate	
ownership-voucher	

Reference: RFC XXXX

Tag: YYY

Name: Zero Touch Information

Returns up to three zero touch bootstrapping artifacts.

Code	Len
XXX	n
zerotouch-information	
owner-certificate	
ownership-voucher	

Reference: RFC XXXX

## 12.2. The IETF XML Registry

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

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

### 12.3. The YANG Module Names Registry

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

name:	ietf-zerotouch-information
namespace:	urn:ietf:params:xml:ns:yang:ietf-zerotouch
prefix:	zt
reference:	RFC XXXX
name:	ietf-zerotouch-bootstrap-server
namespace:	urn:ietf:params:xml:ns:yang:ietf-zerotouch
prefix:	zt
reference:	RFC XXXX

### 13. Other Considerations

Both this document and [I-D.ietf-anima-bootstrapping-keyinfra] define bootstrapping mechanisms. The authors have collaborated on both solutions and believe that each solution has merit and, in fact, can work together. That is, it is possible for a device to support both solutions simultaneously.

### 14. Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by last name): David Harrington, Michael Behringer, Dean Bogdanovic, Martin Bjorklund, Joe Clarke, Toerless Eckert, Stephen Farrell, Ian Farrer, Stephen Hanna, Wes Hardaker, Russ Mundy, Reinaldo Penno, Randy Presuhn, Max Pritikin, Michael Richardson, Phil Shafer, Juergen Schoenwaelder.

Special thanks goes to Steve Hanna, Russ Mundy, and Wes Hardaker for brainstorming the original I-D's solution during the IETF 87 meeting in Berlin.

### 15. References

#### 15.1. Normative References

[I-D.ietf-anima-voucher]  
Watsen, K., Richardson, M., Pritikin, M., and T. Eckert,  
"Voucher and Voucher Revocation Profiles for Bootstrapping  
Protocols", draft-ietf-anima-voucher-00 (work in  
progress), January 2017.

- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<http://www.rfc-editor.org/info/rfc1035>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC2315] Kaliski, B., "PKCS #7: Cryptographic Message Syntax Version 1.5", RFC 2315, DOI 10.17487/RFC2315, March 1998, <<http://www.rfc-editor.org/info/rfc2315>>.
- [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, <<http://www.rfc-editor.org/info/rfc5280>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<http://www.rfc-editor.org/info/rfc6020>>.
- [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, <<http://www.rfc-editor.org/info/rfc6125>>.
- [RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <<http://www.rfc-editor.org/info/rfc6234>>.
- [RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <<http://www.rfc-editor.org/info/rfc6762>>.
- [RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <<http://www.rfc-editor.org/info/rfc6763>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<http://www.rfc-editor.org/info/rfc6991>>.

- [RFC7468] Josefsson, S. and S. Leonard, "Textual Encodings of PKIX, PKCS, and CMS Structures", RFC 7468, DOI 10.17487/RFC7468, April 2015, <<http://www.rfc-editor.org/info/rfc7468>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<http://www.rfc-editor.org/info/rfc8040>>.
- [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>>.

## 15.2. Informative References

- [I-D.ietf-anima-bootstrapping-keyinfra]  
Pritikin, M., Richardson, M., Behringer, M., Bjarnason, S., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructures (BRSKI)", draft-ietf-anima-bootstrapping-keyinfra-04 (work in progress), October 2016.
- [I-D.ietf-netconf-netconf-client-server]  
Watsen, K., Wu, G., and J. Schoenwaelder, "NETCONF Client and Server Models", draft-ietf-netconf-netconf-client-server-01 (work in progress), November 2016.
- [RFC2939] Droms, R., "Procedures and IANA Guidelines for Definition of New DHCP Options and Message Types", BCP 43, RFC 2939, DOI 10.17487/RFC2939, September 2000, <<http://www.rfc-editor.org/info/rfc2939>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<http://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, <<http://www.rfc-editor.org/info/rfc6241>>.
- [RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, DOI 10.17487/RFC6698, August 2012, <<http://www.rfc-editor.org/info/rfc6698>>.



- [RFC6960] Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 6960, DOI 10.17487/RFC6960, June 2013, <<http://www.rfc-editor.org/info/rfc6960>>.
- [RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, DOI 10.17487/RFC7317, August 2014, <<http://www.rfc-editor.org/info/rfc7317>>.
- [RFC8071] Watsen, K., "NETCONF Call Home and RESTCONF Call Home", RFC 8071, DOI 10.17487/RFC8071, February 2017, <<http://www.rfc-editor.org/info/rfc8071>>.

## Appendix A. Change Log

## A.1. ID to 00

- o Major structural update; the essence is the same. Most every section was rewritten to some degree.
- o Added a Use Cases section
- o Added diagrams for "Actors and Roles" and "NMS Precondition" sections, and greatly improved the "Device Boot Sequence" diagram
- o Removed support for physical presence or any ability for configlets to not be signed.
- o Defined the Zero Touch Information DHCP option
- o Added an ability for devices to also download images from configuration servers
- o Added an ability for configlets to be encrypted
- o Now configuration servers only have to support HTTP/S - no other schemes possible

## A.2. 00 to 01

- o Added boot-image and validate-owner annotations to the "Actors and Roles" diagram.
- o Fixed 2nd paragraph in section 7.1 to reflect current use of anyxml.
- o Added encrypted and signed-encrypted examples
- o Replaced YANG module with XSD schema
- o Added IANA request for the Zero Touch Information DHCP Option
- o Added IANA request for media types for boot-image and configuration

## A.3. 01 to 02

- o Replaced the need for a configuration signer with the ability for each NMS to be able to sign its own configurations, using manufacturer signed ownership vouchers and owner certificates.

- o Renamed configuration server to bootstrap server, a more representative name given the information devices download from it.
- o Replaced the concept of a configlet by defining a southbound interface for the bootstrap server using YANG.
- o Removed the IANA request for the boot-image and configuration media types

#### A.4. 02 to 03

- o Minor update, mostly just to add an Editor's Note to show how this draft might integrate with the draft-pritikin-anima-bootstrapping-keyinfra.

#### A.5. 03 to 04

- o Major update formally introducing unsigned data and support for Internet-based redirect servers.
- o Added many terms to Terminology section.
- o Added all new "Guiding Principles" section.
- o Added all new "Sources for Bootstrapping Data" section.
- o Rewrote the "Interactions" section and renamed it "Workflow Overview".

#### A.6. 04 to 05

- o Semi-major update, refactoring the document into more logical parts
- o Created new section for information types
- o Added support for DNS servers
- o Now allows provisional TLS connections
- o Bootstrapping data now supports scripts
- o Device Details section overhauled
- o Security Considerations expanded
- o Filled in enumerations for notification types

## A.7. 05 to 06

- o Minor update
- o Added many Normative and Informative references.
- o Added new section Other Considerations.

## A.8. 06 to 07

- o Minor update
- o Added an Editorial Note section for RFC Editor.
- o Updated the IANA Considerations section.

## A.9. 07 to 08

- o Minor update
- o Updated to reflect review from Michael Richardson.

## A.10. 08 to 09

- o Added in missing "Signature" artifact example.
- o Added recommendation for manufacturers to use interoperable formats and file naming conventions for removable storage devices.
- o Added configuration-handling leaf to guide if config should be merged, replaced, or processed like an edit-config/yang-patch document.
- o Added a pre-configuration script, in addition to the post-configuration script from -05 (issue #15).

## A.11. 09 to 10

- o Factored ownership voucher and voucher revocation to a separate document: draft-kwatsen-netconf-voucher. (issue #11)
- o Removed <configuration-handling> options 'edit-config' and yang-patch'. (issue #12)
- o Defined how a signature over signed-data returned from a bootstrap server is processed. (issue #13)

- o Added recommendation for removable storage devices to use open/standard file systems when possible. (issue #14)
- o Replaced notifications "script-[warning/error]" with "[pre/post]-script-[warning/error]". (goes with issue #15)
- o switched owner-certificate to be encoded using the pkcs#7 format. (issue #16)
- o Replaced md5/sha1 with sha256 inside a choice statement, for future extensibility. (issue #17)
- o A ton of editorial changes, as I went thru the entire draft with a fine-toothed comb.

## A.12. 10 to 11

- o fixed yang validation issues found by IETFYANGPageCompilation. note: these issues were NOT found by pyang --ietf or by the submission-time validator...
- o fixed a typo in the yang module, someone the config false statement was removed.

## A.13. 11 to 12

- o fixed typo that prevented Appendix B from loading the examples correctly.
- o fixed more yang validation issues found by IETFYANGPageCompilation. note: again, these issues were NOT found by pyang --ietf or by the submission-time validator...
- o updated a few of the notification enumerations to be more consistent with the other enumerations (following the warning/error pattern).
- o updated the information-type artifact to state how it's encoded, matching the language that was in Appendix B.

## A.14. 12 to 13

- o defined a standalone artifact to encode the old information-type into a PKCS#7 structure.
- o standalone information artifact hardcodes JSON encoding (to match the voucher draft).

- o combined the information and signature PKCS#7 structures into a single PKCS#7 structure.
- o moved the certificate-revocations into the owner-certificate's PKCS#7 structure.
- o eliminated support for voucher-revocations, to reflect the voucher-draft's switch from revocations to renewals.

Authors' Addresses

Kent Watsen  
Juniper Networks

EMail: kwatsen@juniper.net

Mikael Abrahamsson  
T-Systems

EMail: "mikael.abrahamsson@t-systems.se

NETCONF Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: July 19, 2019

K. Watsen  
Juniper Networks  
M. Abrahamsson  
T-Systems  
I. Farrer  
Deutsche Telekom AG  
January 15, 2019

Secure Zero Touch Provisioning (SZTP)  
draft-ietf-netconf-zerotouch-29

Abstract

This draft presents a technique to securely provision a networking device when it is booting in a factory-default state. Variations in the solution enables it to be used on both public and private networks. The provisioning steps are able to update the boot image, commit an initial configuration, and execute arbitrary scripts to address auxiliary needs. The updated device is subsequently able to establish secure connections with other systems. For instance, a device may establish NETCONF (RFC 6241) and/or RESTCONF (RFC 8040) connections with deployment-specific network management systems.

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 the IANA Considerations section contains placeholder values for DHCP options pending IANA assignment. Please apply the following replacements:

- o "TBD1" --> the assigned value for id-ct-sztpConveyedInfoXML
- o "TBD2" --> the assigned value for id-ct-sztpConveyedInfoJSON
- o "TBD\_IANA\_URL" --> the assigned URL for the IANA registry

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

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

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

- o "2019-01-15" --> the publication date of this draft

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

- o Appendix D. 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 July 19, 2019.

#### Copyright Notice

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

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

#### Table of Contents

1. Introduction . . . . .	5
1.1. Use Cases . . . . .	5
1.2. Terminology . . . . .	6
1.3. Requirements Language . . . . .	8
1.4. Tree Diagrams . . . . .	8
2. Types of Conveyed Information . . . . .	8
2.1. Redirect Information . . . . .	8



2.2. Onboarding Information . . . . .	9
3. Artifacts . . . . .	10
3.1. Conveyed Information . . . . .	10
3.2. Owner Certificate . . . . .	11
3.3. Ownership Voucher . . . . .	12
3.4. Artifact Encryption . . . . .	13
3.5. Artifact Groupings . . . . .	13
4. Sources of Bootstrapping Data . . . . .	14
4.1. Removable Storage . . . . .	15
4.2. DNS Server . . . . .	16
4.3. DHCP Server . . . . .	19
4.4. Bootstrap Server . . . . .	20
5. Device Details . . . . .	21
5.1. Initial State . . . . .	21
5.2. Boot Sequence . . . . .	23
5.3. Processing a Source of Bootstrapping Data . . . . .	25
5.4. Validating Signed Data . . . . .	26
5.5. Processing Redirect Information . . . . .	27
5.6. Processing Onboarding Information . . . . .	28
6. The Conveyed Information Data Model . . . . .	31
6.1. Data Model Overview . . . . .	31
6.2. Example Usage . . . . .	32
6.3. YANG Module . . . . .	34
7. The SZTP Bootstrap Server API . . . . .	40
7.1. API Overview . . . . .	40
7.2. Example Usage . . . . .	41
7.3. YANG Module . . . . .	44
8. DHCP Options . . . . .	56
8.1. DHCPv4 SZTP Redirect Option . . . . .	56
8.2. DHCPv6 SZTP Redirect Option . . . . .	57
8.3. Common Field Encoding . . . . .	58
9. Security Considerations . . . . .	59
9.1. Clock Sensitivity . . . . .	59
9.2. Use of IDevID Certificates . . . . .	59
9.3. Immutable Storage for Trust Anchors . . . . .	59
9.4. Secure Storage for Long-lived Private Keys . . . . .	59
9.5. Blindly Authenticating a Bootstrap Server . . . . .	60
9.6. Disclosing Information to Untrusted Servers . . . . .	60
9.7. Sequencing Sources of Bootstrapping Data . . . . .	61
9.8. Safety of Private Keys used for Trust . . . . .	61
9.9. Increased Reliance on Manufacturers . . . . .	62
9.10. Concerns with Trusted Bootstrap Servers . . . . .	62
9.11. Validity Period for Conveyed Information . . . . .	63
9.12. Cascading Trust via Redirects . . . . .	64
9.13. Possible Reuse of Private Keys . . . . .	64
9.14. Non-Issue with Encrypting Signed Artifacts . . . . .	65
9.15. The "ietf-sztp-conveyed-info" YANG Module . . . . .	65
9.16. The "ietf-sztp-bootstrap-server" YANG Module . . . . .	66

10. IANA Considerations . . . . .	66
10.1. The IETF XML Registry . . . . .	66
10.2. The YANG Module Names Registry . . . . .	67
10.3. The SMI Security for S/MIME CMS Content Type Registry . . . . .	67
10.4. The BOOTP Manufacturer Extensions and DHCP Options Registry . . . . .	67
10.5. The Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Registry . . . . .	68
10.6. The Service Name and Transport Protocol Port Number Registry . . . . .	68
10.7. The DNS Underscore Global Scoped Entry Registry . . . . .	68
11. References . . . . .	69
11.1. Normative References . . . . .	69
11.2. Informative References . . . . .	71
Appendix A. Example Device Data Model . . . . .	74
A.1. Data Model Overview . . . . .	74
A.2. Example Usage . . . . .	74
A.3. YANG Module . . . . .	75
Appendix B. Promoting a Connection from Untrusted to Trusted . . . . .	78
Appendix C. Workflow Overview . . . . .	80
C.1. Enrollment and Ordering Devices . . . . .	80
C.2. Owner Stages the Network for Bootstrap . . . . .	82
C.3. Device Powers On . . . . .	84
Appendix D. Change Log . . . . .	87
D.1. ID to 00 . . . . .	87
D.2. 00 to 01 . . . . .	87
D.3. 01 to 02 . . . . .	87
D.4. 02 to 03 . . . . .	88
D.5. 03 to 04 . . . . .	88
D.6. 04 to 05 . . . . .	88
D.7. 05 to 06 . . . . .	89
D.8. 06 to 07 . . . . .	89
D.9. 07 to 08 . . . . .	89
D.10. 08 to 09 . . . . .	89
D.11. 09 to 10 . . . . .	89
D.12. 10 to 11 . . . . .	90
D.13. 11 to 12 . . . . .	90
D.14. 12 to 13 . . . . .	90
D.15. 13 to 14 . . . . .	91
D.16. 14 to 15 . . . . .	91
D.17. 15 to 16 . . . . .	91
D.18. 16 to 17 . . . . .	92
D.19. 17 to 18 . . . . .	92
D.20. 18 to 19 . . . . .	93
D.21. 19 to 20 . . . . .	93
D.22. 20 to 21 . . . . .	94
D.23. 21 to 22 . . . . .	94
D.24. 22 to 23 . . . . .	94

D.25. 23 to 24	95
D.26. 24 to 25	95
D.27. 25 to 26	96
D.28. 26 to 27	96
D.29. 27 to 28	97
Acknowledgements	97
Authors' Addresses	97

## 1. Introduction

A fundamental business requirement for any network operator is to reduce costs where possible. For network operators, deploying devices to many locations can be a significant cost, as sending trained specialists to each site for installations is both cost prohibitive and does not scale.

This document defines Secure Zero Touch Provisioning (SZTP), a bootstrapping strategy enabling devices to securely obtain bootstrapping data with no installer action beyond physical placement and connecting network and power cables. As such, SZTP enables non-technical personnel to bring up devices in remote locations without the need for any operator input.

The SZTP solution includes updating the boot image, committing an initial configuration, and executing arbitrary scripts to address auxiliary needs. The updated device is subsequently able to establish secure connections with other systems. For instance, a device may establish NETCONF [RFC8040] and/or RESTCONF [RFC6241] connections with deployment-specific network management systems.

This document primarily regards physical devices, where the setting of the device's initial state, described in Section 5.1, occurs during the device's manufacturing process. The SZTP solution may be extended to support virtual machines or other such logical constructs, but details for how this can be accomplished is left for future work.

### 1.1. Use Cases

#### o Device connecting to a remotely administered network

This use-case involves scenarios, such as a remote branch office or convenience store, whereby a device connects as an access gateway to an ISP's network. Assuming it is not possible to customize the ISP's network to provide any bootstrapping support, and with no other nearby device to leverage, the device has no recourse but to reach out to an Internet-based bootstrap server to bootstrap from.

- o Device connecting to a locally administered network

This use-case covers all other scenarios and differs only in that the device may additionally leverage nearby devices, which may direct it to use a local service to bootstrap from. If no such information is available, or the device is unable to use the information provided, it can then reach out to the network just as it would for the remotely administered network use-case.

Conceptual workflows for how SZTP might be deployed are provided in Appendix C.

## 1.2. Terminology

This document uses the following terms (sorted by name):

**Artifact:** The term "artifact" is used throughout to represent any of the three artifacts defined in Section 3 (conveyed information, ownership voucher, and owner certificate). These artifacts collectively provide all the bootstrapping data a device may use.

**Bootstrapping Data:** The term "bootstrapping data" is used throughout this document to refer to the collection of data that a device may obtain during the bootstrapping process. Specifically, it refers to the three artifacts conveyed information, owner certificate, and ownership voucher, as described in Section 3.

**Bootstrap Server:** The term "bootstrap server" is used within this document to mean any RESTCONF server implementing the YANG module defined in Section 7.3.

**Conveyed Information:** The term "conveyed information" is used herein to refer either redirect information or onboarding information. Conveyed information is one of the three bootstrapping artifacts described in Section 3.

**Device:** The term "device" is used throughout this document to refer to a network element that needs to be bootstrapped. See Section 5 for more information about devices.

**Manufacturer:** The term "manufacturer" is used herein to refer to the manufacturer of a device or a delegate of the manufacturer.

**Network Management System (NMS):** The acronym "NMS" is used throughout this document to refer to the deployment-specific management system that the bootstrapping process is responsible for introducing devices to. From a device's perspective, when

the bootstrapping process has completed, the NMS is a NETCONF or RESTCONF client.

**Onboarding Information:** The term "onboarding information" is used herein to refer to one of the two types of "conveyed information" defined in this document, the other being "redirect information". Onboarding information is formally defined by the "onboarding-information" YANG-data structure in Section 6.3.

**Onboarding Server:** The term "onboarding server" is used herein to refer to a bootstrap server that only returns onboarding information.

**Owner:** The term "owner" is used throughout this document to refer to the person or organization that purchased or otherwise owns a device.

**Owner Certificate:** The term "owner certificate" is used in this document to represent an X.509 certificate that binds an owner identity to a public key, which a device can use to validate a signature over the conveyed information artifact. The owner certificate may be communicated along with its chain of intermediate certificates leading up to a known trust anchor. The owner certificate is one of the three bootstrapping artifacts described in Section 3.

**Ownership Voucher:** The term "ownership voucher" is used in this document to represent the voucher artifact defined in [RFC8366]. The ownership voucher is used to assign a device to an owner. The ownership voucher is one of the three bootstrapping artifacts described in Section 3.

**Redirect Information:** The term "redirect information" is used herein to refer to one of the two types of "conveyed information" defined in this document, the other being "onboarding information". Redirect information is formally defined by the "redirect-information" YANG-data structure in Section 6.3.

**Redirect Server:** The term "redirect server" is used to refer to a bootstrap server that only returns redirect information. A redirect server is particularly useful when hosted by a manufacturer, as a well-known (e.g., Internet-based) resource to redirect devices to deployment-specific bootstrap servers.

**Signed Data:** The term "signed data" is used throughout to mean conveyed information that has been signed, specifically by a private key possessed by a device's owner.

**Unsigned Data:** The term "unsigned data" is used throughout to mean conveyed information that has not been signed.

### 1.3. 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.

### 1.4. Tree Diagrams

Tree diagrams used in this document follow the notation defined in [RFC8340].

## 2. Types of Conveyed Information

This document defines two types of conveyed information that devices can access during the bootstrapping process. These conveyed information types are described in this section. Examples are provided in Section 6.2

### 2.1. Redirect Information

Redirect information redirects a device to another bootstrap server. Redirect information encodes a list of bootstrap servers, each specifying the bootstrap server's hostname (or IP address), an optional port, and an optional trust anchor certificate that the device can use to authenticate the bootstrap server with.

Redirect information is YANG modeled data formally defined by the "redirect-information" container in the YANG module presented in Section 6.3. This container has the tree diagram shown below.

```
+--:(redirect-information)
  +-- redirect-information
    +-- bootstrap-server* [address]
      +-- address          inet:host
      +-- port?           inet:port-number
      +-- trust-anchor?   cms
```

Redirect information may be trusted or untrusted. The redirect information is trusted whenever it is obtained via a secure connection to a trusted bootstrap server, or whenever it is signed by the device's owner. In all other cases, the redirect information is untrusted.

Trusted redirect information is useful for enabling a device to establish a secure connection to a specified bootstrap server, which is possible when the redirect information includes the bootstrap server's trust anchor certificate.

Untrusted redirect information is useful for directing a device to a bootstrap server where signed data has been staged for it to obtain. Note that, when the redirect information is untrusted, devices discard any potentially included trust anchor certificates.

How devices process redirect information is described in Section 5.5.

## 2.2. Onboarding Information

Onboarding information provides data necessary for a device to bootstrap itself and establish secure connections with other systems. As defined in this document, onboarding information can specify details about the boot image a device must be running, specify an initial configuration the device must commit, and specify scripts that the device must successfully execute.

Onboarding information is YANG modeled data formally defined by the "onboarding-information" container in the YANG module presented in Section 6.3. This container has the tree diagram shown below.

```
+--:(onboarding-information)
  +-- onboarding-information
    +-- boot-image
      +-- os-name?          string
      +-- os-version?      string
      +-- download-uri*    inet:uri
      +-- image-verification* [hash-algorithm]
        +-- hash-algorithm identityref
        +-- hash-value     yang:hex-string
    +-- configuration-handling? enumeration
    +-- pre-configuration-script? script
    +-- configuration?      binary
    +-- post-configuration-script? script
```

Onboarding information must be trusted for it to be of any use to a device. There is no option for a device to process untrusted onboarding information.

Onboarding information is trusted whenever it is obtained via a secure connection to a trusted bootstrap server, or whenever it is signed by the device's owner. In all other cases, the onboarding information is untrusted.

How devices process onboarding information is described in Section 5.6.

### 3. Artifacts

This document defines three artifacts that can be made available to devices while they are bootstrapping. Each source of bootstrapping data specifies how it provides the artifacts defined in this section (see Section 4).

#### 3.1. Conveyed Information

The conveyed information artifact encodes the essential bootstrapping data for the device. This artifact is used to encode the redirect information and onboarding information types discussed in Section 2.

The conveyed information artifact is a CMS structure, as described in [RFC5652], encoded using ASN.1 distinguished encoding rules (DER), as specified in ITU-T X.690 [ITU.X690.2015]. The CMS structure MUST contain content conforming to the YANG module specified in Section 6.3.

The conveyed information CMS structure may encode signed or unsigned bootstrapping data. When the bootstrapping data is signed, it may also be encrypted but, from a terminology perspective, it is still "signed data" Section 1.2.

When the conveyed information artifact is unsigned, as it might be when communicated over trusted channels, the CMS structure's top-most content type MUST be one of the OIDs described in Section 10.3 (i.e., id-ct-sztpConveyedInfoXML or id-ct-sztpConveyedInfoJSON), or the OID id-data (1.2.840.113549.1.7.1). When the OID id-data is used, the encoding (JSON, XML, etc.) SHOULD be communicated externally. In either case, the associated content is an octet string containing "conveyed-information" data in the expected encoding.

When the conveyed information artifact is unsigned and encrypted, as it might be when communicated over trusted channels but, for some reason, the operator wants to ensure that only the device is able to see the contents, the CMS structure's top-most content type MUST be the OID id-envelopedData (1.2.840.113549.1.7.3). Furthermore, the encryptedContentInfo's content type MUST be one of the OIDs described in Section 10.3 (i.e., id-ct-sztpConveyedInfoXML or id-ct-sztpConveyedInfoJSON), or the OID id-data (1.2.840.113549.1.7.1). When the OID id-data is used, the encoding (JSON, XML, etc.) SHOULD be communicated externally. In either case, the associated content is an octet string containing "conveyed-information" data in the expected encoding.



When the conveyed information artifact is signed, as it might be when communicated over untrusted channels, the CMS structure's top-most content type MUST be the OID id-signedData (1.2.840.113549.1.7.2). Furthermore, the inner eContentType MUST be one of the OIDs described in Section 10.3 (i.e., id-ct-sztpConveyedInfoXML or id-ct-sztpConveyedInfoJSON), or the OID id-data (1.2.840.113549.1.7.1). When the OID id-data is used, the encoding (JSON, XML, etc.) SHOULD be communicated externally. In either case, the associated content or eContent is an octet string containing "conveyed-information" data in the expected encoding.

When the conveyed information artifact is signed and encrypted, as it might be when communicated over untrusted channels and privacy is important, the CMS structure's top-most content type MUST be the OID id-envelopedData (1.2.840.113549.1.7.3). Furthermore, the encryptedContentInfo's content type MUST be the OID id-signedData (1.2.840.113549.1.7.2), whose eContentType MUST be one of the OIDs described in Section 10.3 (i.e., id-ct-sztpConveyedInfoXML or id-ct-sztpConveyedInfoJSON), or the OID id-data (1.2.840.113549.1.7.1). When the OID id-data is used, the encoding (JSON, XML, etc.) SHOULD be communicated externally. In either case, the associated content or eContent is an octet string containing "conveyed-information" data in the expected encoding.

### 3.2. Owner Certificate

The owner certificate artifact is an X.509 certificate [RFC5280] that is used to identify an "owner" (e.g., an organization). The owner certificate can be signed by any certificate authority (CA). The owner certificate either MUST have no Key Usage specified or the Key Usage MUST at least set the "digitalSignature" bit. The values for the owner certificate's "subject" and/or "subjectAltName" are not constrained by this document.

The owner certificate is used by a device to verify the signature over the conveyed information artifact (Section 3.1) that the device should have also received, as described in Section 3.5. In particular, the device verifies the signature using the public key in the owner certificate over the content contained within the conveyed information artifact.

The owner certificate artifact is formally a CMS structure, as specified by [RFC5652], encoded using ASN.1 distinguished encoding rules (DER), as specified in ITU-T X.690 [ITU.X690.2015].

The owner certificate CMS structure MUST contain the owner certificate itself, as well as all intermediate certificates leading to the "pinned-domain-cert" certificate specified in the ownership

voucher. The owner certificate artifact MAY optionally include the "pinned-domain-cert" as well.

In order to support devices deployed on private networks, the owner certificate CMS structure MAY also contain suitably fresh, as determined by local policy, revocation objects (e.g., CRLs). Having these revocation objects stapled to the owner certificate may obviate the need for the device to have to download them dynamically using the CRL distribution point or an OCSP responder specified in the associated certificates.

When unencrypted, the owner certificate artifact's CMS structure's top-most content type MUST be the OID id-signedData (1.2.840.113549.1.7.2). The inner SignedData structure is the degenerate form, whereby there are no signers, that is commonly used to disseminate certificates and revocation objects.

When encrypted, the owner certificate artifact's CMS structure's top-most content type MUST be the OID id-envelopedData (1.2.840.113549.1.7.3), and the encryptedContentInfo's content type MUST be the OID id-signedData (1.2.840.113549.1.7.2), whereby the inner SignedData structure is the degenerate form that has no signers commonly used to disseminate certificates and revocation objects.

### 3.3. Ownership Voucher

The ownership voucher artifact is used to securely identify a device's owner, as it is known to the manufacturer. The ownership voucher is signed by the device's manufacturer.

The ownership voucher is used to verify the owner certificate (Section 3.2) that the device should have also received, as described in Section 3.5. In particular, the device verifies that the owner certificate has a chain of trust leading to the trusted certificate included in the ownership voucher ("pinned-domain-cert"). Note that this relationship holds even when the owner certificate is a self-signed certificate, and hence also the pinned-domain-cert.

When unencrypted, the ownership voucher artifact is as defined in [RFC8366]. As described, it is a CMS structure whose top-most content type MUST be the OID id-signedData (1.2.840.113549.1.7.2), whose eContentType MUST be OID id-ct-animaJSONVoucher (1.2.840.113549.1.9.16.1), or the OID id-data (1.2.840.113549.1.7.1). When the OID id-data is used, the encoding (JSON, XML, etc.) SHOULD be communicated externally. In either case, the associated content is an octet string containing ietf-voucher data in the expected encoding.

When encrypted, the ownership voucher artifact's CMS structure's top-most content type MUST be the OID id-envelopedData (1.2.840.113549.1.7.3), and the encryptedContentInfo's content type MUST be the OID id-signedData (1.2.840.113549.1.7.2), whose eContentType MUST be OID id-ct-animaJSONVoucher (1.2.840.113549.1.9.16.1), or the OID id-data (1.2.840.113549.1.7.1). When the OID id-data is used, the encoding (JSON, XML, etc.) SHOULD be communicated externally. In either case, the associated content is an octet string containing ietf-voucher data in the expected encoding.

### 3.4. Artifact Encryption

Each of the three artifacts MAY be individually encrypted. Encryption may be important in some environments where the content is considered sensitive.

Each of the three artifacts are encrypted in the same way, by the unencrypted form being encapsulated inside a CMS EnvelopedData type.

As a consequence, both the conveyed information and ownership voucher artifacts are signed and then encrypted, never encrypted and then signed.

This sequencing has the advantage of shrouding the signer's certificate, and ensuring that the owner knows the content being signed. This sequencing further enables the owner to inspect an unencrypted voucher obtained from a manufacturer and then encrypt the voucher later themselves, perhaps while also stapling in current revocation objects, when ready to place the artifact in an unsafe location.

When encrypted, the CMS MUST be encrypted using a secure device identity certificate for the device. This certificate MAY be the same as the TLS-level client certificate the device uses when connecting to bootstrap servers. The owner must possess the device's identity certificate at the time of encrypting the data. How the owner comes to possess the device's identity certificate for this purpose is outside the scope of this document.

### 3.5. Artifact Groupings

The previous sections discussed the bootstrapping artifacts, but only certain groupings of these artifacts make sense to return in the various bootstrapping situations described in this document. These groupings are:

Unsigned Data: This artifact grouping is useful for cases when transport level security can be used to convey trust (e.g., HTTPS), or when the conveyed information can be processed in a provisional manner (i.e. unsigned redirect information).

Signed Data, without revocations: This artifact grouping is useful when signed data is needed (i.e., because the data is obtained from an untrusted source and it cannot be processed provisionally) and either revocations are not needed or the revocations can be obtained dynamically.

Signed Data, with revocations: This artifact grouping is useful when signed data is needed (i.e., because the data is obtained from an untrusted source and it cannot be processed provisionally), and revocations are needed, and the revocations cannot be obtained dynamically.

The presence of each artifact, and any distinguishing characteristics, are identified for each artifact grouping in the table below ("yes/no" regards if the artifact is present in the artifact grouping):

Artifact Grouping	Conveyed Information	Ownership Voucher	Owner Certificate
Unsigned Data	Yes, no sig	No	No
Signed Data, without revocations	Yes, with sig	Yes, without revocations	Yes, without revocations
Signed Data, with revocations	Yes, with sig	Yes, with revocations	Yes, with revocations

#### 4. Sources of Bootstrapping Data

This section defines some sources for bootstrapping data that a device can access. The list of sources defined here is not meant to be exhaustive. It is left to future documents to define additional sources for obtaining bootstrapping data.

For each source of bootstrapping data defined in this section, details are given for how the three artifacts listed in Section 3 are provided.

#### 4.1. Removable Storage

A directly attached removable storage device (e.g., a USB flash drive) MAY be used as a source of SZTP bootstrapping data.

Use of a removable storage device is compelling, as it does not require any external infrastructure to work. It is notable that the raw boot image file can also be located on the removable storage device, enabling a removable storage device to be a fully self-standing bootstrapping solution.

To use a removable storage device as a source of bootstrapping data, a device need only detect if the removable storage device is plugged in and mount its filesystem.

A removable storage device is an untrusted source of bootstrapping data. This means that the information stored on the removable storage device either MUST be signed or MUST be information that can be processed provisionally (e.g., unsigned redirect information).

From an artifact perspective, since a removable storage device presents itself as a filesystem, the bootstrapping artifacts need to be presented as files. The three artifacts defined in Section 3 are mapped to files below.

##### Artifact to File Mapping:

Conveyed Information: Mapped to a file containing the binary artifact described in Section 3.1 (e.g., conveyed-information.cms).

Owner Certificate: Mapped to a file containing the binary artifact described in Section 3.2 (e.g., owner-certificate.cms).

Ownership Voucher: Mapped to a file containing the binary artifact described in Section 3.3 (e.g., ownership-voucher.cms or ownership-voucher.vcj).

The format of the removable storage device's filesystem and the naming of the files are outside the scope of this document. However, in order to facilitate interoperability, it is RECOMMENDED devices support open and/or standards based filesystems. It is also RECOMMENDED that devices assume a file naming convention that enables more than one instance of bootstrapping data (i.e., for different devices) to exist on a removable storage device. The file naming convention SHOULD additionally be unique to the manufacturer, in

order to enable bootstrapping data from multiple manufacturers to exist on a removable storage device.

#### 4.2. DNS Server

A DNS server MAY be used as a source of SZTP bootstrapping data.

Using a DNS server may be a compelling option for deployments having existing DNS infrastructure, as it enables a touchless bootstrapping option that does not entail utilizing an Internet based resource hosted by a 3rd-party.

DNS is an untrusted source of bootstrapping data. Even if DNSSEC [RFC6698] is used to authenticate the various DNS resource records (e.g., A, AAAA, CERT, TXT, and TLSA), the device cannot be sure that the domain returned to it from e.g., a DHCP server, belongs to its rightful owner. This means that the information stored in the DNS records either MUST be signed (per this document, not DNSSEC), or MUST be information that can be processed provisionally (e.g., unsigned redirect information).

##### 4.2.1. DNS Queries

Devices claiming to support DNS as a source of bootstrapping data MUST first query for device-specific DNS records and, only if doing so does not result in a successful bootstrap, then MUST query for device-independent DNS records.

For each of the device-specific and device-independent queries, devices MUST first query using multicast DNS [RFC6762] and, only if doing so does not result in a successful bootstrap, then MUST query again using unicast DNS [RFC1035] [RFC7766], assuming the address of a DNS server is known, such as it may be using techniques similar to those described in Section 11 of [RFC6763], which is referenced a few times in this document, even though this document does not itself use DNS-SD (RFC 6763 is identified herein as an Informative reference).

When querying for device-specific DNS records, devices MUST query for TXT records [RFC1035] under "<serial-number>.\_sztp", where <serial-number> is the device's serial number (the same value as in the device's secure device identity certificate), and "\_sztp" is the globally scoped DNS attribute registered by this document in Section 10.7.

Example device-specific DNS record queries:

```
TXT in <serial-number>._sztp.local. (multicast)
TXT in <serial-number>._sztp.<domain>. (unicast)
```

When querying for device-independent DNS records, devices MUST query for SRV records [RFC2782] under "\_sztp.\_tcp", where "\_sztp" is the service name registered by this document in Section 10.6, and "\_tcp" is the globally scoped DNS attribute registered by [I-D.ietf-dnsop-attrleaf].

Note that a device-independent response is anyway only able to encode unsigned data, since signed data necessitates the use of a device-specific ownership voucher. Use of SRV records maximally leverages existing DNS standards. A response containing multiple SRV records is comparable to an unsigned redirect information's list of bootstrap servers.

Example device-independent DNS record queries:

```
SRV in _sztp._tcp.local. (multicast)
SRV in _sztp._tcp.<domain>. (unicast)
```

#### 4.2.2. DNS Response for Device-Specific Queries

For device-specific queries, the three bootstrapping artifacts defined in Section 3 are encoded into the TXT records using key/value pairs, similar to the technique described in Section 6.3 of [RFC6763].

Artifact to TXT Record Mapping:

Conveyed Information: Mapped to a TXT record having the key "ci" and the value being the binary artifact described in Section 3.1.

Owner Certificate: Mapped to a TXT record having the key "oc" and the value being the binary artifact described in Section 3.2.

Ownership Voucher: Mapped to a TXT record having the key "ov" and the value being the binary artifact described in Section 3.3.

Devices MUST ignore any other keys that may be returned.

Note that, despite the name, TXT records can and SHOULD (per Section 6.5 of [RFC6763]) encode binary data.

Following is an example of a device-specific response, as it might be presented by a user-agent, containing signed data. This example assumes that the device's serial number is "<serial-number>", the domain is "example.com", and that "<binary data>" represents the binary artifact:

```
<serial-number>._sztp.example.com. 3600 IN TXT "ci=<binary data>"
<serial-number>._sztp.example.com. 3600 IN TXT "oc=<binary data>"
<serial-number>._sztp.example.com. 3600 IN TXT "ov=<binary data>"
```

Note that, in the case that "ci" encodes unsigned data, the "oc" and "ov" keys would not be present in the response.

#### 4.2.3. DNS Response for Device-Independent Queries

For device-independent queries, the three bootstrapping artifacts defined in Section 3 are encoded into the SVR records as follows.

Artifact to SRV Record Mapping:

Conveyed Information: This artifact is not supported directly. Instead, the essence of unsigned redirect information is mapped to SVR records per [RFC2782].

Owner Certificate: Not supported. Device-independent responses are never encode signed data, and hence there is no need for an owner certificate artifact.

Ownership Voucher: Not supported. Device-independent responses are never encode signed data, and hence there is no need for an ownership voucher artifact.

Following is an example of a device-independent response, as it might be presented by a user-agent, containing (effectively) unsigned redirect information to four bootstrap servers. This example assumes that the domain is "example.com" and that there are four bootstrap servers "sztp[1-4]":

```
_sztp._tcp.example.com. 1800 IN SRV 0 0 443 sztp1.example.com.
_sztpt._tcp.example.com. 1800 IN SRV 1 0 443 sztp2.example.com.
_sztpt._tcp.example.com. 1800 IN SRV 2 0 443 sztp3.example.com.
_sztpt._tcp.example.com. 1800 IN SRV 2 0 443 sztp4.example.com.
```

Note that, in this example, "sztp3" and "sztp4" have equal priority, and hence effectively represent a clustered pair of bootstrap servers. While "sztp1" and "sztp2" only have a single SRV record each, it may be that the record points to a load-balancer fronting a cluster of bootstrap servers.

While this document does not use DNS-SD [RFC6763], per Section 12.2 of that RFC, mDNS responses SHOULD also include all address records (type "A" and "AAAA") named in the SRV rdata.



#### 4.2.4. Size of Signed Data

The signed data artifacts are large by DNS conventions. In the smallest-footprint scenario, they are each a few kilobytes in size. However, onboarding information can easily be several kilobytes in size, and has the potential to be many kilobytes in size.

All resource records, including TXT records, have an upper size limit of 65535 bytes, since "RDLENGTH" is a 16-bit field (Section 3.2.1 in [RFC1035]). If it is ever desired to encode onboarding information that exceeds this limit, the DNS records returned should instead encode redirect information, to direct the device to a bootstrap server from which the onboarding information can be obtained.

Given the expected size of the TXT records, it is unlikely that signed data will fit into a UDP-based DNS packet, even with the EDNS(0) Extensions [RFC6891] enabled. Depending on content, signed data may also not fit into a multicast DNS packet, which bounds the size to 9000 bytes, per Section 17 in [RFC6762]. Thus it is expected that DNS Transport over TCP [RFC7766] will be required in order to return signed data.

#### 4.3. DHCP Server

A DHCP server MAY be used as a source of SZTP bootstrapping data.

Using a DHCP server may be a compelling option for deployments having existing DHCP infrastructure, as it enables a touchless bootstrapping option that does not entail utilizing an Internet based resource hosted by a 3rd-party.

A DHCP server is an untrusted source of bootstrapping data. Thus the information stored on the DHCP server either MUST be signed, or it MUST be information that can be processed provisionally (e.g., unsigned redirect information).

However, unlike other sources of bootstrapping data described in this document, the DHCP protocol (especially DHCP for IPv4) is very limited in the amount of data that can be conveyed, to the extent that signed data cannot be communicated. This means that only unsigned redirect information can be conveyed via DHCP.

Since the redirect information is unsigned, it SHOULD NOT include the optional trust anchor certificate, as it takes up space in the DHCP message, and the device would have to discard it anyway. For this reason, the DHCP options defined in Section 8 do not enable the trust anchor certificate to be encoded.

From an artifact perspective, the three artifacts defined in Section 3 are mapped to the DHCP fields specified in Section 8 as follows.

#### Artifact to DHCP Option Fields Mapping:

Conveyed Information: This artifact is not supported directly. Instead, the essence of unsigned redirect information is mapped to the DHCP options described in Section 8.

Owner Certificate: Not supported. There is not enough space in the DHCP packet to hold an owner certificate artifact.

Ownership Voucher: Not supported. There is not enough space in the DHCP packet to hold an ownership voucher artifact.

#### 4.4. Bootstrap Server

A bootstrap server MAY be used as a source of SZTP bootstrapping data. A bootstrap server is defined as a RESTCONF [RFC8040] server implementing the YANG module provided in Section 7.

Using a bootstrap server as a source of bootstrapping data is a compelling option as it MAY use transport-level security, obviating the need for signed data, which may be easier to deploy in some situations.

Unlike any other source of bootstrapping data described in this document, a bootstrap server is not only a source of data, but it can also receive data from devices using the YANG-defined "report-progress" RPC defined in the YANG module (Section 7.3). The "report-progress" RPC enables visibility into the bootstrapping process (e.g., warnings and errors), and provides potentially useful information upon completion (e.g., the device's SSH host-keys).

A bootstrap server may be a trusted or an untrusted source of bootstrapping data, depending on if the device learned about the bootstrap server's trust anchor from a trusted source. When a bootstrap server is trusted, the conveyed information returned from it MAY be signed. When the bootstrap server is untrusted, the conveyed information either MUST be signed or MUST be information that can be processed provisionally (e.g., unsigned redirect information).

From an artifact perspective, since a bootstrap server presents data conforming to a YANG data model, the bootstrapping artifacts need to be mapped to YANG nodes. The three artifacts defined in Section 3

are mapped to "output" nodes of the "get-bootstrapping-data" RPC defined in Section 7.3 below.

Artifact to Bootstrap Server Mapping:

Conveyed Information: Mapped to the "conveyed-information" leaf in the output of the "get-bootstrapping-data" RPC.

Owner Certificate: Mapped to the "owner-certificate" leaf in the output of the "get-bootstrapping-data" RPC.

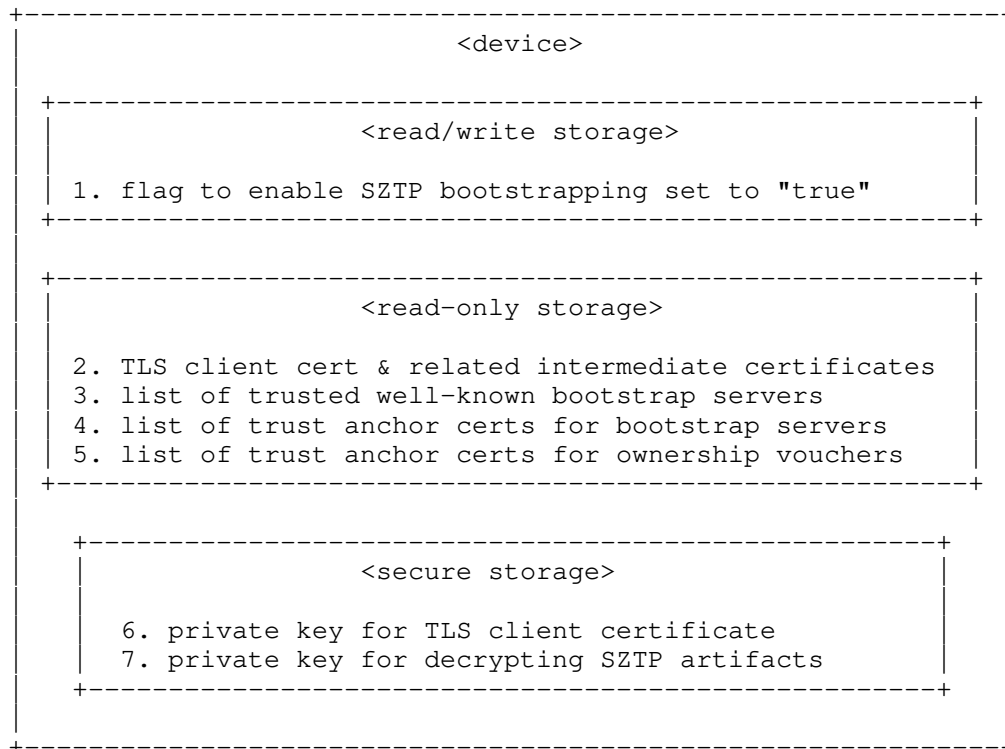
Ownership Voucher: Mapped to the "ownership-voucher" leaf in the output of the "get-bootstrapping-data" RPC.

SZTP bootstrap servers have only two endpoints, one for the "get-bootstrapping-data" RPC and one for the "report-progress" RPC. These RPCs use the authenticated RESTCONF username to isolate the execution of the RPC from other devices.

## 5. Device Details

Devices supporting the bootstrapping strategy described in this document MUST have the preconfigured state and bootstrapping logic described in the following sections.

### 5.1. Initial State



Each numbered item below corresponds to a numbered item in the diagram above.

1. Devices MUST have a configurable variable that is used to enable/disable SZTP bootstrapping. This variable MUST be enabled by default in order for SZTP bootstrapping to run when the device first powers on. Because it is a goal that the configuration installed by the bootstrapping process disables SZTP bootstrapping, and because the configuration may be merged into the existing configuration, using a configuration node that relies on presence is NOT RECOMMENDED, as it cannot be removed by the merging process.
2. Devices that support loading bootstrapping data from bootstrap servers (see Section 4.4) SHOULD possess a TLS-level client certificate and any intermediate certificates leading to the certificate's well-known trust-anchor. The well-known trust anchor certificate may be an intermediate certificate or a self-signed root certificate. To support devices not having a client certificate, devices MAY, alternatively or in addition to, identify and authenticate themselves to the bootstrap server

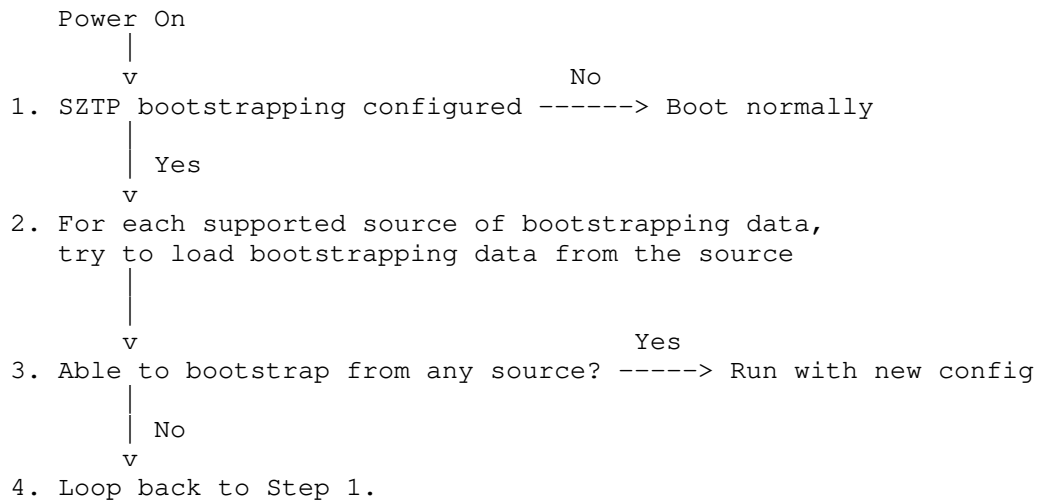
using an HTTP authentication scheme, as allowed by Section 2.5 in [RFC8040]; however, this document does not define a mechanism for operator input enabling, for example, the entering of a password.

3. Devices that support loading bootstrapping data from well-known bootstrap servers MUST possess a list of the well-known bootstrap servers. Consistent with redirect information (Section 2.1, each bootstrap server can be identified by its hostname or IP address, and an optional port.
4. Devices that support loading bootstrapping data from well-known bootstrap servers MUST also possess a list of trust anchor certificates that can be used to authenticate the well-known bootstrap servers. For each trust anchor certificate, if it is not itself a self-signed root certificate, the device SHOULD also possess the chain of intermediate certificates leading up to and including the self-signed root certificate.
5. Devices that support loading signed data (see Section 1.2) MUST possess the trust anchor certificates for validating ownership vouchers. For each trust anchor certificate, if it is not itself a self-signed root certificate, the device SHOULD also possess the chain of intermediate certificates leading up to and including the self-signed root certificate.
6. Devices that support using a TLS-level client certificate to identify and authenticate themselves to a bootstrap server MUST possess the private key that corresponds to the public key encoded in the TLS-level client certificate. This private key SHOULD be securely stored, ideally in a cryptographic processor, such as a trusted platform module (TPM) chip.
7. Devices that support decrypting SZTP artifacts MUST possess the private key that corresponds to the public key encoded in the secure device identity certificate used when encrypting the artifacts. This private key SHOULD be securely stored, ideally in a cryptographic processor, such as a trusted platform module (TPM) chip. This private key MAY be the same as the one associated to the TLS-level client certificate used when connecting to bootstrap servers.

A YANG module representing this data is provided in Appendix A.

## 5.2. Boot Sequence

A device claiming to support the bootstrapping strategy defined in this document MUST support the boot sequence described in this section.



Note: At any time, the device MAY be configured via an alternate provisioning mechanism (e.g., CLI).

Each numbered item below corresponds to a numbered item in the diagram above.

1. When the device powers on, it first checks to see if SZTP bootstrapping is configured, as is expected to be the case for the device's preconfigured initial state. If SZTP bootstrapping is not configured, then the device boots normally.
2. The device iterates over its list of sources for bootstrapping data (Section 4). Details for how to process a source of bootstrapping data are provided in Section 5.3.
3. If the device is able to bootstrap itself from any of the sources of bootstrapping data, it runs with the new bootstrapped configuration.
4. Otherwise the device MUST loop back through the list of bootstrapping sources again.

This document does not limit the simultaneous use of alternate provisioning mechanisms. Such mechanisms may include, for instance, a command line interface (CLI), a web-based user interface, or even another bootstrapping protocol. Regardless how it is configured, the configuration SHOULD unset the flag enabling SZTP bootstrapping discussed in Section 5.1.

### 5.3. Processing a Source of Bootstrapping Data

This section describes a recursive algorithm that devices can use to, ultimately, obtain onboarding information. The algorithm is recursive because sources of bootstrapping data may return redirect information, which causes the algorithm to run again, for the newly discovered sources of bootstrapping data. An expression that captures all possible successful sequences of bootstrapping data is: zero or more redirect information responses, followed by one onboarding information response.

An important aspect of the algorithm is knowing when data needs to be signed or not. The following figure provides a summary of options:

Kind of Bootstrapping Data	Untrusted Source Can Provide?	Trusted Source Can Provide?
Unsigned Redirect Info	Yes+	Yes
Signed Redirect Info	Yes	Yes*
Unsigned Onboarding Info	No	Yes
Signed Onboarding Info	Yes	Yes*

The '+' above denotes that the source redirected to MUST return signed data, or more unsigned redirect information.

The '\*' above denotes that, while possible, it is generally unnecessary for a trusted source to return signed data.

The recursive algorithm uses a conceptual global-scoped variable called "trust-state". The trust-state variable is initialized to FALSE. The ultimate goal of this algorithm is for the device to process onboarding information (Section 2.2) while the trust-state variable is TRUE.

If the source of bootstrapping data (Section 4) is a bootstrap server (Section 4.4), and the device is able to authenticate the bootstrap server using X.509 certificate path validation ([RFC6125], Section 6) to one of the device's preconfigured trust anchors, or to a trust anchor that it learned from a previous step, then the device MUST set trust-state to TRUE.

When establishing a connection to a bootstrap server, whether trusted or untrusted, the device MUST identify and authenticate itself to the bootstrap server using a TLS-level client certificate and/or an HTTP authentication scheme, per Section 2.5 in [RFC8040]. If both authentication mechanisms are used, they MUST both identify the same serial number.

When sending a client certificate, the device MUST also send all of the intermediate certificates leading up to, and optionally including, the client certificate's well-known trust anchor certificate.

For any source of bootstrapping data (e.g., Section 4), if any artifact obtained is encrypted, the device MUST first decrypt it using the private key associated with the device certificate used to encrypt the artifact.

If the conveyed information artifact is signed, and the device is able to validate the signed data using the algorithm described in Section 5.4, then the device MUST set trust-state to TRUE; otherwise, if the device is unable to validate the signed data, the device MUST set trust-state to FALSE. Note, this is worded to cover the special case when signed data is returned even from a trusted source of bootstrapping data.

If the conveyed information artifact contains redirect information, the device MUST, within limits of how many recursive loops the device allows, process the redirect information as described in Section 5.5. Implementations MUST limit the maximum number of recursive redirects allowed; the maximum number of recursive redirects allowed SHOULD be no more than ten. This is the recursion step, it will cause the device to reenter this algorithm, but this time the data source will definitely be a bootstrap server, as redirect information is only able to redirect devices to bootstrap servers.

If the conveyed information artifact contains onboarding information, and trust-state is FALSE, the device MUST exit the recursive algorithm (as this is not allowed, see the figure above), returning to the bootstrapping sequence described in Section 5.2. Otherwise, the device MUST attempt to process the onboarding information as described in Section 5.6. Whether the processing of the onboarding information succeeds or fails, the device MUST exit the recursive algorithm, returning to the bootstrapping sequence described in Section 5.2, the only difference being in how it responds to the "Able to bootstrap from any source?" conditional described in the figure in the section.

#### 5.4. Validating Signed Data

Whenever a device is presented signed data, it MUST validate the signed data as described in this section. This includes the case where the signed data is provided by a trusted source.

Whenever there is signed data, the device MUST also be provided an ownership voucher and an owner certificate. How all the needed



artifacts are provided for each source of bootstrapping data is described in Section 4.

In order to validate signed data, the device MUST first authenticate the ownership voucher by validating its signature to one of its preconfigured trust anchors (see Section 5.1), which may entail using additional intermediate certificates attached to the ownership voucher. If the device has an accurate clock, it MUST verify that the ownership voucher was created in the past (i.e., "created-on" < now) and, if the "expires-on" leaf is present, the device MUST verify that the ownership voucher has not yet expired (i.e., now < "expires-on"). The device MUST verify that the ownership voucher's "assertion" value is acceptable (e.g., some devices may only accept the assertion value "verified"). The device MUST verify that the ownership voucher specifies the device's serial number in the "serial-number" leaf. If the "idevid-issuer" leaf is present, the device MUST verify that the value is set correctly. If the authentication of the ownership voucher is successful, the device extracts the "pinned-domain-cert" node, an X.509 certificate, that is needed to verify the owner certificate in the next step.

The device MUST next authenticate the owner certificate by performing X.509 certificate path verification to the trusted certificate extracted from the ownership voucher's "pinned-domain-cert" node. This verification may entail using additional intermediate certificates attached to the owner certificate artifact. If the ownership voucher's "domain-cert-revocation-checks" node's value is set to "true", the device MUST verify the revocation status of the certificate chain used to sign the owner certificate and, if suitably-fresh revocation status is unattainable or if it is determined that a certificate has been revoked, the device MUST NOT validate the owner certificate.

Finally, the device MUST verify that the conveyed information artifact was signed by the validated owner certificate.

If any of these steps fail, the device MUST invalidate the signed data and not perform any subsequent steps.

#### 5.5. Processing Redirect Information

In order to process redirect information (Section 2.1), the device MUST follow the steps presented in this section.

Processing redirect information is straightforward; the device sequentially steps through the list of provided bootstrap servers until it can find one it can bootstrap from.

If a hostname is provided, and the hostname's DNS resolution is to more than one IP address, the device MUST attempt to connect to all of the DNS resolved addresses at least once, before moving on to the next bootstrap server. If the device is able to obtain bootstrapping data from any of the DNS resolved addresses, it MUST immediately process that data, without attempting to connect to any of the other DNS resolved addresses.

If the redirect information is trusted (e.g., trust-state is TRUE), and the bootstrap server entry contains a trust anchor certificate, then the device MUST authenticate the specified bootstrap server's TLS server certificate using X.509 certificate path validation ([RFC6125], Section 6) to the specified trust anchor. If the bootstrap server entry does not contain a trust anchor certificate device, the device MUST establish a provisional connection to the bootstrap server (i.e., by blindly accepting its server certificate), and set trust-state to FALSE.

If the redirect information is untrusted (e.g., trust-state is FALSE), the device MUST discard any trust anchors provided by the redirect information and establish a provisional connection to the bootstrap server (i.e., by blindly accepting its TLS server certificate).

#### 5.6. Processing Onboarding Information

In order to process onboarding information (Section 2.2), the device MUST follow the steps presented in this section.

When processing onboarding information, the device MUST first process the boot image information (if any), then execute the pre-configuration script (if any), then commit the initial configuration (if any), and then execute the post-configuration script (if any), in that order.

When the onboarding information is obtained from a trusted bootstrap server, the device MUST send the "bootstrap-initiated" progress report, and send either a terminating "boot-image-installed-rebooting", "bootstrap-complete", or error specific progress report. If the bootstrap server's "get-bootstrapping-data" RPC-reply's "reporting-level" node is set to "verbose", the device MUST additionally send all appropriate non-terminating progress reports (e.g., initiated, warning, complete, etc.). Regardless of the reporting-level indicated by the bootstrap server, the device MAY send progress reports beyond the mandatory ones specified for the given reporting level.

When the onboarding information is obtained from an untrusted bootstrap server, the device MUST NOT send any progress reports to the bootstrap server, even though the onboarding information was, necessarily, signed and authenticated. Please be aware that bootstrap servers are recommended to promote untrusted connections to trusted connections, in the last paragraph of Section 9.6, so as to, in part, be able to collect progress reports from devices.

If the device encounters an error at any step, it MUST stop processing the onboarding information and return to the bootstrapping sequence described in Section 5.2. In the context of a recursive algorithm, the device MUST return to the enclosing loop, not back to the very beginning. Some state MAY be retained from the bootstrapping process (e.g., updated boot image, logs, remnants from a script, etc.). However, the retained state MUST NOT be active in any way (e.g., no new configuration or running of software), and MUST NOT hinder the ability for the device to continue the bootstrapping sequence (i.e., process onboarding information from another bootstrap server).

At this point, the specific ordered sequence of actions the device MUST perform is described.

If the onboarding information is obtained from a trusted bootstrap server, the device MUST send a "bootstrap-initiated" progress report. It is an error if the device does not receive back the "204 No Content" HTTP status line. If an error occurs, the device MUST try to send a "bootstrap-error" progress report before exiting.

The device MUST parse the provided onboarding information document, to extract values used in subsequent steps. Whether using a stream-based parser or not, if there is an error when parsing the onboarding information, and the device is connected to a trusted bootstrap server, the device MUST try to send a "parsing-error" progress report before exiting.

If boot image criteria are specified, the device MUST first determine if the boot image it is running satisfies the specified boot image criteria. If the device is already running the specified boot image, then it skips the remainder of this step. If the device is not running the specified boot image, then it MUST download, verify, and install, in that order, the specified boot image, and then reboot. If connected to a trusted bootstrap server, the device MAY try to send a "boot-image-mismatch" progress report. To download the boot image, the device MUST only use the URIs supplied by the onboarding information. To verify the boot image, the device MUST either use one of the verification fingerprints supplied by the onboarding information, or use a cryptographic signature embedded into the boot

image itself using a mechanism not described by this document. Before rebooting, if connected to a trusted bootstrap server, the device MUST try to send a "boot-image-installed-rebooting" progress report. Upon rebooting, the bootstrapping process runs again, which will eventually come to this step again, but then the device will be running the specified boot image, and thus will move to processing the next step. If an error occurs at any step while the device is connected to a trusted bootstrap server (i.e., before the reboot), the device MUST try to send a "boot-image-error" progress report before exiting.

If a pre-configuration script has been specified, the device MUST execute the script, capture any output emitted from the script, and check if the script had any warnings or errors. If an error occurs while the device is connected to a trusted bootstrap server, the device MUST try to send a "pre-script-error" progress report before exiting.

If an initial configuration has been specified, the device MUST atomically commit the provided initial configuration, using the approach specified by the "configuration-handling" leaf. If an error occurs while the device is connected to a trusted bootstrap server, the device MUST try to send a "config-error" progress report before exiting.

If a post-configuration script has been specified, the device MUST execute the script, capture any output emitted from the script, and check if the script had any warnings or errors. If an error occurs while the device is connected to a trusted bootstrap server, the device MUST try to send a "post-script-error" progress report before exiting.

If the onboarding information was obtained from a trusted bootstrap server, and the result of the bootstrapping process did not disable the "flag to enable SZTP bootstrapping" described in Section 5.1, the device SHOULD send an "bootstrap-warning" progress report.

If the onboarding information was obtained from a trusted bootstrap server, the device MUST send a "bootstrap-complete" progress report. It is an error if the device does not receive back the "204 No Content" HTTP status line. If an error occurs, the device MUST try to send a "bootstrap-error" progress report before exiting.

At this point, the device has completely processed the bootstrapping data.

The device is now running its initial configuration. Notably, if NETCONF Call Home or RESTCONF Call Home [RFC8071] is configured, the

device initiates trying to establish the call home connections at this time.

#### Implementation Notes:

Implementations may vary in how to ensure no unwanted state is retained when an error occurs.

Following are some guidelines for if the implementation chooses to undo previous steps:

- \* When an error occurs, the device must rollback the current step and any previous steps.
- \* Most steps are atomic. For example, the processing of a configuration is specified above as atomic, and the processing of scripts is similarly specified as atomic in the "ietf-sztp-conveyed-info" YANG module.
- \* In case the error occurs after the initial configuration was committed, the device must restore the configuration to the configuration that existed prior to the configuration being committed.
- \* In case the error occurs after a script had executed successfully, it may be helpful for the implementation to define scripts as being able to take a conceptual input parameter indicating that the script should remove its previously set state.

## 6. The Conveyed Information Data Model

This section defines a YANG 1.1 [RFC7950] module that is used to define the data model for the conveyed information artifact described in Section 3.1. This data model uses the "yang-data" extension statement defined in [RFC8040]. Examples illustrating this data model are provided in Section 6.2.

### 6.1. Data Model Overview

The following tree diagram provides an overview of the data model for the conveyed information artifact.

```
module: ietf-sztp-conveyed-info

yang-data conveyed-information:
  +-- (information-type)
  +--:(redirect-information)
  |   +-- redirect-information
  |   |   +-- bootstrap-server* [address]
  |   |   |   +-- address          inet:host
  |   |   |   +-- port?           inet:port-number
  |   |   |   +-- trust-anchor?   cms
  |   +--:(onboarding-information)
  |   |   +-- onboarding-information
  |   |   |   +-- boot-image
  |   |   |   |   +-- os-name?          string
  |   |   |   |   +-- os-version?       string
  |   |   |   |   +-- download-uri*     inet:uri
  |   |   |   |   +-- image-verification* [hash-algorithm]
  |   |   |   |   |   +-- hash-algorithm identityref
  |   |   |   |   |   +-- hash-value    yang:hex-string
  |   |   |   +-- configuration-handling? enumeration
  |   |   +-- pre-configuration-script? script
  |   +-- configuration? binary
  +-- post-configuration-script? script
```

## 6.2. Example Usage

The following example illustrates how redirect information (Section 2.1) can be encoded using JSON.

```
{
  "ietf-sztp-conveyed-info:redirect-information" : {
    "bootstrap-server" : [
      {
        "address" : "sztp1.example.com",
        "port" : 8443,
        "trust-anchor" : "base64encodedvalue=="
      },
      {
        "address" : "sztp2.example.com",
        "port" : 8443,
        "trust-anchor" : "base64encodedvalue=="
      },
      {
        "address" : "sztp3.example.com",
        "port" : 8443,
        "trust-anchor" : "base64encodedvalue=="
      }
    ]
  }
}
```

The following example illustrates how onboarding information (Section 2.2) can be encoded using JSON.

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

```
{
  "ietf-sztp-conveyed-info:onboarding-information" : {
    "boot-image" : {
      "os-name" : "VendorOS",
      "os-version" : "17.2R1.6",
      "download-uri" : [ "http://some/path/to/raw/file" ],
      "image-verification" : [
        {
          "hash-algorithm" : "ietf-sztp-conveyed-info:sha-256",
          "hash-value" : "ba:ec:cf:a5:67:82:b4:10:77:c6:67:a6:22:ab:\
7d:50:04:a7:8b:8f:0e:db:02:8b:f4:75:55:fb:c1:13:b2:33"
        }
      ]
    },
    "configuration-handling" : "merge",
    "pre-configuration-script" : "base64encodedvalue==",
    "configuration" : "base64encodedvalue==",
    "post-configuration-script" : "base64encodedvalue=="
  }
}
```

### 6.3. YANG Module

The conveyed information data model is defined by the YANG module presented in this section.

This module uses data types defined in [RFC5280], [RFC5652], [RFC6234], and [RFC6991], an extension statement from [RFC8040], and an encoding defined in [ITU.X690.2015].

```
<CODE BEGINS> file "ietf-sztp-conveyed-info@2019-01-15.yang"
module ietf-sztp-conveyed-info {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-sztp-conveyed-info";
  prefix sztp-info;

  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-restconf {
    prefix rc;
    reference "RFC 8040: RESTCONF Protocol";
  }

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

  contact
    "WG Web:  http://tools.ietf.org/wg/netconf
    WG List:  <mailto:netconf@ietf.org>
    Author:   Kent Watsen <mailto:kwatsen@juniper.net>";

  description
    "This module defines the data model for the Conveyed
    Information artifact defined in RFC XXXX: Secure Zero Touch
    Provisioning (SZTP).

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



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

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

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

```
revision 2019-01-15 {
  description
    "Initial version";
  reference
    "RFC XXXX: Secure Zero Touch Provisioning (SZTP)";
}

// identities

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

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

// typedefs

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).";
}
```

```
}

// yang-data

rc:yang-data "conveyed-information" {
  choice information-type {
    mandatory true;
    description
      "This choice statement ensures the response contains
       redirect-information or onboarding-information.";
    container redirect-information {
      description
        "Redirect information is described in Section 2.1 in
         RFC XXXX. Its purpose is to redirect a device to
         another bootstrap server.";
      reference
        "RFC XXXX: Secure Zero Touch Provisioning (SZTP)";
      list bootstrap-server {
        key "address";
        min-elements 1;
        description
          "A bootstrap server entry.";
        leaf address {
          type inet:host;
          mandatory true;
          description
            "The IP address or hostname of the bootstrap server the
             device should redirect to.";
        }
        leaf port {
          type inet:port-number;
          default "443";
          description
            "The port number the bootstrap server listens on. If no
             port is specified, the IANA-assigned port for 'https'
             (443) is used.";
        }
      }
      leaf trust-anchor {
        type cms;
        description
          "A CMS structure that MUST contain the chain of
           X.509 certificates needed to authenticate the TLS
           certificate presented by this bootstrap server.

           The CMS MUST only contain a single chain of
           certificates. The bootstrap server 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 bootstrap server's TLS certificate, only one certificate is present.

If needed by the device, this CMS structure MAY also contain suitably fresh 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.";
}
}
}
container onboarding-information {
  description
    "Onboarding information is described in Section 2.2 in
    RFC XXXX. Its purpose is to provide the device everything
    it needs to bootstrap itself.";
  reference
    "RFC XXXX: Secure Zero Touch Provisioning (SZTP)";
  container boot-image {
    description
      "Specifies criteria for the boot image the device MUST
      be running, as well as information enabling the device
      to install the required boot image.";
    leaf os-name {
      type string;
      description
        "The name of the operating system software the device
        MUST be running in order to not require a software
        image upgrade (ex. VendorOS).";
    }
    leaf os-version {
      type string;
      description
        "The version of the operating system software the
        device MUST be running in order to not require a
        software image upgrade (ex. 17.3R2.1).";
    }
    leaf-list download-uri {
```

```
type inet:uri;
ordered-by user;
description
  "An ordered list of URIs to where the same boot image
  file may be obtained. How the URI schemes (http, ftp,
  etc.) a device supports are known is vendor specific.
  If a secure scheme (e.g., https) is provided, a device
  MAY establish an untrusted connection to the remote
  server, by blindly accepting the server's end-entity
  certificate, to obtain the boot image.";
}
list image-verification {
  must '../download-uri' {
    description
      "Download URIs must be provided if an image is to
      be verified.";
  }
  key hash-algorithm;
  description
    "A list of hash values that a device can use to verify
    boot image files with.";
  leaf hash-algorithm {
    type identityref {
      base "hash-algorithm";
    }
    description
      "Identifies the hash algorithm used.";
  }
  leaf hash-value {
    type yang:hex-string;
    mandatory true;
    description
      "The hex-encoded value of the specified hash
      algorithm over the contents of the boot image
      file.";
  }
}
}
leaf configuration-handling {
  type enumeration {
    enum "merge" {
      description
        "Merge configuration into the running datastore.";
    }
    enum "replace" {
      description
        "Replace the existing running datastore with the
        passed configuration.";
    }
  }
}
```

```
    }
  }
  must '../configuration';
  description
    "This enumeration indicates how the server should process
    the provided configuration.";
}
leaf pre-configuration-script {
  type script;
  description
    "A script that, when present, is executed before the
    configuration has been processed.";
}
leaf configuration {
  type binary;
  must '../configuration-handling';
  description
    "Any configuration known to the device. The use of
    the 'binary' type enables e.g., XML-content to be
    embedded into a JSON document. The exact encoding
    of the content, as with the scripts, is vendor
    specific.";
}
leaf post-configuration-script {
  type script;
  description
    "A script that, when present, is executed after the
    configuration has been processed.";
}
}
}
```

```
typedef script {
  type binary;
  description
    "A device specific script that enables the execution of
    commands to perform actions not possible thru configuration
    alone.
```

No attempt is made to standardize the contents, running context, or programming language of the script, other than that it can indicate if any warnings or errors occurred and can emit output. The contents of the script are considered specific to the vendor, product line, and/or model of the device.

If the script execution indicates that an warning occurred,

then the device MUST assume that the script had a soft error that the script believes will not affect manageability.

If the script execution indicates that an error occurred, the device MUST assume the script had a hard error that the script believes will affect manageability. In this case, the script is required to gracefully exit, removing any state that might hinder the device's ability to continue the bootstrapping sequence (e.g., process onboarding information obtained from another bootstrap server).";

```
    }  
  }  
<CODE ENDS>
```

## 7. The SZTP Bootstrap Server API

This section defines the API for bootstrap servers. The API is defined as that produced by a RESTCONF [RFC8040] server that supports the YANG 1.1 [RFC7950] module defined in this section.

### 7.1. API Overview

The following tree diagram provides an overview for the bootstrap server RESTCONF API.

```
module: ietf-sztp-bootstrap-server
```

```
rpcs:
  +---x get-bootstrapping-data
  |   +---w input
  |   |   +---w signed-data-preferred?    empty
  |   |   +---w hw-model?                  string
  |   |   +---w os-name?                   string
  |   |   +---w os-version?                string
  |   |   +---w nonce?                     binary
  |   +---ro output
  |   |   +---ro reporting-level?          enumeration {onboarding-server}?
  |   |   +---ro conveyed-information      cms
  |   |   +---ro owner-certificate?        cms
  |   |   +---ro ownership-voucher?        cms
  +---x report-progress {onboarding-server}?
  |   +---w input
  |   |   +---w progress-type              enumeration
  |   |   +---w message?                   string
  |   |   +---w ssh-host-keys
  |   |   |   +---w ssh-host-key* []
  |   |   |   |   +---w algorithm          string
  |   |   |   |   +---w key-data          binary
  |   |   +---w trust-anchor-certs
  |   |   |   +---w trust-anchor-cert*    cms
```

## 7.2. Example Usage

This section presents three examples illustrating the bootstrap server's API. Two examples are provided for the "get-bootstrapping-data" RPC (once to an untrusted bootstrap server, and again to a trusted bootstrap server), and one example for the "report-progress" RPC.

The following example illustrates a device using the API to fetch its bootstrapping data from an untrusted bootstrap server. In this example, the device sends the "signed-data-preferred" input parameter and receives signed data in the response.

## REQUEST

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

```
POST /restconf/operations/ietf-sztp-bootstrap-server:get-bootstrapi\
ng-data HTTP/1.1
HOST: example.com
Content-Type: application/yang.data+xml
```

```
<input
  xmlns="urn:ietf:params:xml:ns:yang:ietf-sztp-bootstrap-server">
  <signed-data-preferred/>
</input>
```

## RESPONSE

```
HTTP/1.1 200 OK
Date: Sat, 31 Oct 2015 17:02:40 GMT
Server: example-server
Content-Type: application/yang.data+xml
```

```
<output
  xmlns="urn:ietf:params:xml:ns:yang:ietf-sztp-bootstrap-server">
  <conveyed-information>base64encodedvalue==</conveyed-information>
  <owner-certificate>base64encodedvalue==</owner-certificate>
  <ownership-voucher>base64encodedvalue==</ownership-voucher>
</output>
```

The following example illustrates a device using the API to fetch its bootstrapping data from a trusted bootstrap server. In this example, the device sends addition input parameters to the bootstrap server, which it may use when formulating its response to the device.



## REQUEST

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

```
POST /restconf/operations/ietf-sztp-bootstrap-server:get-bootstrapi\
ng-data HTTP/1.1
HOST: example.com
Content-Type: application/yang.data+xml
```

```
<input
  xmlns="urn:ietf:params:xml:ns:yang:ietf-sztp-bootstrap-server">
  <hw-model>model-x</hw-model>
  <os-name>vendor-os</os-name>
  <os-version>17.3R2.1</os-version>
  <nonce>extralongbase64encodedvalue=</nonce>
</input>
```

## RESPONSE

```
HTTP/1.1 200 OK
Date: Sat, 31 Oct 2015 17:02:40 GMT
Server: example-server
Content-Type: application/yang.data+xml
```

```
<output
  xmlns="urn:ietf:params:xml:ns:yang:ietf-sztp-bootstrap-server">
  <reporting-level>verbose</reporting-level>
  <conveyed-information>base64encodedvalue==</conveyed-information>
</output>
```

The following example illustrates a device using the API to post a progress report to a bootstrap server. Illustrated below is the "bootstrap-complete" message, but the device may send other progress reports to the server while bootstrapping. In this example, the device is sending both its SSH host keys and a TLS server certificate, which the bootstrap server may, for example, pass to an NMS, as discussed in Appendix C.3.

## REQUEST

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

```
POST /restconf/operations/ietf-sztp-bootstrap-server:report-progress\
HTTP/1.1
HOST: example.com
Content-Type: application/yang.data+xml
```

```
<input
  xmlns="urn:ietf:params:xml:ns:yang:ietf-sztp-bootstrap-server">
  <progress-type>bootstrap-complete</progress-type>
  <message>example message</message>
  <ssh-host-keys>
    <ssh-host-key>
      <algorithm>ssh-rsa</algorithm>
      <key-data>base64encodedvalue==</key-data>
    </ssh-host-key>
    <ssh-host-key>
      <algorithm>rsa-sha2-256</algorithm>
      <key-data>base64encodedvalue==</key-data>
    </ssh-host-key>
  </ssh-host-keys>
  <trust-anchor-certs>
    <trust-anchor-cert>base64encodedvalue==</trust-anchor-cert>
  </trust-anchor-certs>
</input>
```

## RESPONSE

```
HTTP/1.1 204 No Content
Date: Sat, 31 Oct 2015 17:02:40 GMT
Server: example-server
```

### 7.3. YANG Module

The bootstrap server's device-facing API is normatively defined by the YANG module defined in this section.

This module uses data types defined in [RFC4253], [RFC5652], [RFC5280], [RFC6960], and [RFC8366], uses an encoding defined in [ITU.X690.2015], and makes a reference to [RFC4250] and [RFC6187].

```
<CODE BEGINS> file "ietf-sztp-bootstrap-server@2019-01-15.yang"
module ietf-sztp-bootstrap-server {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-sztp-bootstrap-server";
  prefix sztp-svr;
```

## organization

"IETF NETCONF (Network Configuration) Working Group";

## contact

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

WG List: <<mailto:netconf@ietf.org>>

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

## description

"This module defines an interface for bootstrap servers, as defined by RFC XXXX: Secure Zero Touch Provisioning (SZTP).

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

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

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

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

revision 2019-01-15 {

description

"Initial version";

reference

"RFC XXXX: Secure Zero Touch Provisioning (SZTP)";

}

// features

feature redirect-server {

description

"The server supports being a 'redirect server'.";

}

feature onboarding-server {

description

"The server supports being an 'onboarding server'.";

}

```
// typedefs

typedef cms {
    type binary;
    description
        "A CMS 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).";
}

// RPCs

rpc get-bootstrapping-data {
    description
        "This RPC enables a device, as identified by the RESTCONF
        username, to obtain bootstrapping data that has been made
        available for it.";
    input {
        leaf signed-data-preferred {
            type empty;
            description
                "This optional input parameter enables a device to
                communicate to the bootstrap server that it prefers
                to receive signed data. Devices SHOULD always send
                this parameter when the bootstrap server is untrusted.
                Upon receiving this input parameter, the bootstrap
                server MUST return either signed data, or unsigned
                redirect information; the bootstrap server MUST NOT
                return unsigned onboarding information.";
        }
        leaf hw-model {
            type string;
            description
                "This optional input parameter enables a device to
                communicate to the bootstrap server its vendor specific
                hardware model number. This parameter may be needed,
                for instance, when a device's IDevID certificate does
                not include the 'hardwareModelName' value in its
                subjectAltName field, as is allowed by 802.1AR-2009.";
            reference

```

```
        "IEEE 802.1AR-2009: IEEE Standard for Local and
          metropolitan area networks - Secure Device Identity";
    }
    leaf os-name {
        type string;
        description
            "This optional input parameter enables a device to
             communicate to the bootstrap server the name of its
             operating system. This parameter may be useful if
             the device, as identified by its serial number, can
             run more than one type of operating system (e.g.,
             on a white-box system.";
    }
    leaf os-version {
        type string;
        description
            "This optional input parameter enables a device to
             communicate to the bootstrap server the version of its
             operating system. This parameter may be used by a
             bootstrap server to return an operating system specific
             response to the device, thus negating the need for a
             potentially expensive boot-image update.";
    }
    leaf nonce {
        type binary {
            length "16..32";
        }
        description
            "This optional input parameter enables a device to
             communicate to the bootstrap server a nonce value.
             This may be especially useful for devices lacking
             an accurate clock, as then the bootstrap server
             can dynamically obtain from the manufacturer a
             voucher with the nonce value in it, as described
             in RFC 8366.";
        reference
            "RFC 8366:
             A Voucher Artifact for Bootstrapping Protocols";
    }
}
output {
    leaf reporting-level {
        if-feature onboarding-server;
        type enumeration {
            enum standard {
                description
                    "Send just the progress reports required by RFC XXXX.";
                reference

```

```
        "RFC XXXX: Secure Zero Touch Provisioning (SZTP)";
    }
    enum verbose {
        description
            "Send additional progress reports that might help
            troubleshooting an SZTP bootstrapping issue.";
    }
}
default standard;
description
    "Specifies the reporting level for progress reports the
    bootstrap server would like to receive when processing
    onboarding information. Progress reports are not sent
    when processing redirect information, or when the
    bootstrap server is untrusted (e.g., device sent the
    '<signed-data-preferred>' input parameter).";
}
leaf conveyed-information {
    type cms;
    mandatory true;
    description
        "An SZTP conveyed information artifact, as described in
        Section 3.1 of RFC XXXX.";
    reference
        "RFC XXXX: Secure Zero Touch Provisioning (SZTP)";
}
leaf owner-certificate {
    type cms;
    must '../ownership-voucher' {
        description
            "An ownership voucher must be present whenever an owner
            certificate is presented.";
    }
    description
        "An owner certificate artifact, as described in Section
        3.2 of RFC XXXX. This leaf is optional because it is
        only needed when the conveyed information artifact is
        signed.";
    reference
        "RFC XXXX: Secure Zero Touch Provisioning (SZTP)";
}
leaf ownership-voucher {
    type cms;
    must '../owner-certificate' {
        description
            "An owner certificate must be present whenever an
            ownership voucher is presented.";
    }
}
```

```
        description
            "An ownership voucher artifact, as described by Section
            3.3 of RFC XXXX. This leaf is optional because it is
            only needed when the conveyed information artifact is
            signed.";
        reference
            "RFC XXXX: Secure Zero Touch Provisioning (SZTP)";
    }
}

rpc report-progress {
    if-feature onboarding-server;
    description
        "This RPC enables a device, as identified by the RESTCONF
        username, to report its bootstrapping progress to the
        bootstrap server. This RPC is expected to be used when
        the device obtains onboarding-information from a trusted
        bootstrap server.";
    input {
        leaf progress-type {
            type enumeration {
                enum "bootstrap-initiated" {
                    description
                        "Indicates that the device just used the
                        'get-bootstrapping-data' RPC. The 'message' node
                        below MAY contain any additional information that
                        the manufacturer thinks might be useful.";
                }
                enum "parsing-initiated" {
                    description
                        "Indicates that the device is about to start parsing
                        the onboarding information. This progress type is
                        only for when parsing is implemented as a distinct
                        step.";
                }
                enum "parsing-warning" {
                    description
                        "Indicates that the device had a non-fatal error when
                        parsing the response from the bootstrap server. The
                        'message' node below SHOULD indicate the specific
                        warning that occurred.";
                }
                enum "parsing-error" {
                    description
                        "Indicates that the device encountered a fatal error
                        when parsing the response from the bootstrap server.
                        For instance, this could be due to malformed encoding,
```

```
the device expecting signed data when only unsigned
data is provided, the ownership voucher not listing
the device's serial number, or because the signature
didn't match. The 'message' node below SHOULD
indicate the specific error. This progress type
also indicates that the device has abandoned trying
to bootstrap off this bootstrap server.";
}
enum "parsing-complete" {
  description
    "Indicates that the device successfully completed
    parsing the onboarding information. This progress
    type is only for when parsing is implemented as a
    distinct step.";
}
enum "boot-image-initiated" {
  description
    "Indicates that the device is about to start
    processing the boot-image information.";
}
enum "boot-image-warning" {
  description
    "Indicates that the device encountered a non-fatal
    error condition when trying to install a boot-image.
    A possible reason might include a need to reformat a
    partition causing loss of data. The 'message' node
    below SHOULD indicate any warning messages that were
    generated.";
}
enum "boot-image-error" {
  description
    "Indicates that the device encountered an error when
    trying to install a boot-image, which could be for
    reasons such as a file server being unreachable,
    file not found, signature mismatch, etc. The
    'message' node SHOULD indicate the specific error
    that occurred. This progress type also indicates
    that the device has abandoned trying to bootstrap
    off this bootstrap server.";
}
enum "boot-image-mismatch" {
  description
    "Indicates that the device that has determined that
    it is not running the correct boot image. This
    message SHOULD precipitate trying to download
    a boot image.";
}
enum "boot-image-installed-rebooting" {
```



```
description
  "Indicates that the device successfully installed
  a new boot image and is about to reboot. After
  sending this progress type, the device is not
  expected to access the bootstrap server again
  for this bootstrapping attempt.";
}
enum "boot-image-complete" {
  description
    "Indicates that the device believes that it is
    running the correct boot-image.";
}
enum "pre-script-initiated" {
  description
    "Indicates that the device is about to execute the
    'pre-configuration-script'.";
}
enum "pre-script-warning" {
  description
    "Indicates that the device obtained a warning from the
    'pre-configuration-script' when it was executed. The
    'message' node below SHOULD capture any output the
    script produces.";
}
enum "pre-script-error" {
  description
    "Indicates that the device obtained an error from the
    'pre-configuration-script' when it was executed. The
    'message' node below SHOULD capture any output the
    script produces. This progress type also indicates
    that the device has abandoned trying to bootstrap
    off this bootstrap server.";
}
enum "pre-script-complete" {
  description
    "Indicates that the device successfully executed the
    'pre-configuration-script'.";
}
enum "config-initiated" {
  description
    "Indicates that the device is about to commit the
    initial configuration.";
}
enum "config-warning" {
  description
    "Indicates that the device obtained warning messages
    when it committed the initial configuration. The
    'message' node below SHOULD indicate any warning
```

```
        messages that were generated.";
    }
    enum "config-error" {
        description
            "Indicates that the device obtained error messages
            when it committed the initial configuration. The
            'message' node below SHOULD indicate the error
            messages that were generated. This progress type
            also indicates that the device has abandoned trying
            to bootstrap off this bootstrap server.";
    }
    enum "config-complete" {
        description
            "Indicates that the device successfully committed
            the initial configuration.";
    }
    enum "post-script-initiated" {
        description
            "Indicates that the device is about to execute the
            'post-configuration-script'.";
    }
    enum "post-script-warning" {
        description
            "Indicates that the device obtained a warning from the
            'post-configuration-script' when it was executed. The
            'message' node below SHOULD capture any output the
            script produces.";
    }
    enum "post-script-error" {
        description
            "Indicates that the device obtained an error from the
            'post-configuration-script' when it was executed. The
            'message' node below SHOULD capture any output the
            script produces. This progress type also indicates
            that the device has abandoned trying to bootstrap
            off this bootstrap server.";
    }
    enum "post-script-complete" {
        description
            "Indicates that the device successfully executed the
            'post-configuration-script'.";
    }
    enum "bootstrap-warning" {
        description
            "Indicates that a warning condition occurred for which
            there no other 'progress-type' enumeration is deemed
            suitable. The 'message' node below SHOULD describe
            the warning.";
```

```
    }
    enum "bootstrap-error" {
        description
            "Indicates that an error condition occurred for which
            there no other 'progress-type' enumeration is deemed
            suitable. The 'message' node below SHOULD describe
            the error. This progress type also indicates that
            the device has abandoned trying to bootstrap off
            this bootstrap server.";
    }
    enum "bootstrap-complete" {
        description
            "Indicates that the device successfully processed
            all 'onboarding-information' provided, and that it
            is ready to be managed. The 'message' node below
            MAY contain any additional information that the
            manufacturer thinks might be useful. After sending
            this progress type, the device is not expected to
            access the bootstrap server again.";
    }
    enum "informational" {
        description
            "Indicates any additional information not captured
            by any of the other progress types. For instance,
            a message indicating that the device is about to
            reboot after having installed a boot-image could
            be provided. The 'message' node below SHOULD
            contain information that the manufacturer thinks
            might be useful.";
    }
}
mandatory true;
description
    "The type of progress report provided.";
}
leaf message {
    type string;
    description
        "An optional arbitrary value.";
}
container ssh-host-keys {
    when "../progress-type = 'bootstrap-complete'" {
        description
            "SSH host keys are only sent when the progress type
            is 'bootstrap-complete'.";
    }
}
description
    "A list of SSH host keys an NMS may use to authenticate
```

```
        subsequent SSH-based connections to this device (e.g.,
        netconf-ssh, netconf-ch-ssh).";
list ssh-host-key {
  description
    "An SSH host key an NMS may use to authenticate
    subsequent SSH-based connections to this device
    (e.g., netconf-ssh, netconf-ch-ssh).";
  reference
    "RFC 4253: The Secure Shell (SSH) Transport Layer
    Protocol";
  leaf algorithm {
    type string;
    mandatory true;
    description
      "The public key algorithm name for this SSH key.

      Valid values are listed in the 'Public Key Algorithm
      Names' subregistry of the 'Secure Shell (SSH) Protocol
      Parameters' registry maintained by IANA.";
    reference
      "RFC 4250: The Secure Shell (SSH) Protocol Assigned
      Numbers
      IANA URL: https://www.iana.org/assignments/ssh-param\
eters/ssh-parameters.xhtml#ssh-parameters-19
      ('\\" added for formatting reasons)";
  }
  leaf key-data {
    type binary;
    mandatory true;
    description
      "The binary public key data for this SSH key, as
      specified by RFC 4253, Section 6.6, i.e.:

      string      certificate or public key format
                  identifier
      byte[n]     key/certificate data.";
    reference
      "RFC 4253: The Secure Shell (SSH) Transport Layer
      Protocol";
  }
}
}
}
container trust-anchor-certs {
  when "../progress-type = 'bootstrap-complete'" {
    description
      "Trust anchors are only sent when the progress type
      is 'bootstrap-complete'.";
  }
}
```

```
description
  "A list of trust anchor certificates an NMS may use to
  authenticate subsequent certificate-based connections
  to this device (e.g., restconf-tls, netconf-tls, or
  even netconf-ssh with X.509 support from RFC 6187).
  In practice, trust anchors for IDevID certificates do
  not need to be conveyed using this mechanism.";
reference
  "RFC 6187:
    X.509v3 Certificates for Secure Shell Authentication.";
leaf-list trust-anchor-cert {
  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.

    The CMS MUST contain the chain of X.509 certificates
    needed to authenticate the certificate presented by
    the device.

    The CMS MUST contain only a single chain of
    certificates. The last certificate in the chain
    MUST be the issuer for the device's end-entity
    certificate.

    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 device's
    end-entity certificate, only one certificate is
    present.

    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.
    RFC 5652:
      Cryptographic Message Syntax (CMS)";
}
```

```
  }
}
}
}
}
<CODE ENDS>
```

## 8. DHCP Options

This section defines two DHCP options, one for DHCPv4 and one for DHCPv6. These two options are semantically the same, though syntactically different.

### 8.1. DHCPv4 SZTP Redirect Option

The DHCPv4 SZTP Redirect Option is used to provision the client with one or more URIs for bootstrap servers that can be contacted to attempt further configuration.

#### DHCPv4 SZTP Redirect Option

```

      0                                     1
      0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|  option-code (143)  |  option-length  |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
.
.  bootstrap-server-list (variable length)  .
.
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

- \* option-code: OPTION\_V4\_SZTP\_REDIRECT (143)
- \* option-length: The option length in octets.
- \* bootstrap-server-list: A list of servers for the client to attempt contacting, in order to obtain further bootstrapping data, in the format shown in Section 8.3.

#### DHCPv4 Client Behavior

Clients MAY request the OPTION\_V4\_SZTP\_REDIRECT by including its option code in the Parameter Request List (55) in DHCP request messages.

On receipt of a DHCPv4 Reply message which contains the OPTION\_V4\_SZTP\_REDIRECT, the client processes the response according to Section 5.5, with the understanding that the "address" and "port" values are encoded in the URIs.

Any invalid URI entries received in the uri-data field are ignored by the client. If OPTION\_V4\_SZTP\_REDIRECT does not contain at least one valid URI entry in the uri-data field, then the client MUST discard the option.

As the list of URIs may exceed the maximum allowed length of a single DHCPv4 option (255 octets), the client MUST implement [RFC3396], allowing the URI list to be split across a number of OPTION\_V4\_SZTP\_REDIRECT option instances.

#### DHCPv4 Server Behavior

The DHCPv4 server MAY include a single instance of Option OPTION\_V4\_SZTP\_REDIRECT in DHCP messages it sends. Servers MUST NOT send more than one instance of the OPTION\_V4\_SZTP\_REDIRECT option.

The server's DHCP message MUST contain only a single instance of the OPTION\_V4\_SZTP\_REDIRECT's 'bootstrap-server-list' field. However, the list of URIs in this field may exceed the maximum allowed length of a single DHCPv4 option (per [RFC3396]).

If the length of 'bootstrap-server-list' is small enough to fit into a single instance of OPTION\_V4\_SZTP\_REDIRECT, the server MUST NOT send more than one instance of this option.

If the length of the 'bootstrap-server-list' field is too large to fit into a single option, then OPTION\_V4\_SZTP\_REDIRECT MUST be split into multiple instances of the option according to the process described in [RFC3396].

## 8.2. DHCPv6 SZTP Redirect Option

The DHCPv6 SZTP Redirect Option is used to provision the client with one or more URIs for bootstrap servers that can be contacted to attempt further configuration.

#### DHCPv6 SZTP Redirect Option

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|          option-code (136)          |          option-length          |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
.          bootstrap-server-list (variable length)          .
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

- \* option-code: OPTION\_V6\_SZTP\_REDIRECT (136)
- \* option-length: The option length in octets.
- \* bootstrap-server-list: A list of servers for the client to attempt contacting, in order to obtain further bootstrapping data, in the format shown in Section 8.3.

#### DHCPv6 Client Behavior

Clients MAY request the `OPTION_V6_SZTP_REDIRECT` option, as defined in [RFC8415], Sections 18.2.1, 18.2.2, 18.2.4, 18.2.5, 18.2.6, and 21.7.

As a convenience to the reader, we mention here that the client includes requested option codes in the Option Request Option.

On receipt of a DHCPv6 Reply message which contains the `OPTION_V6_SZTP_REDIRECT`, the client processes the response according to Section 5.5, with the understanding that the "address" and "port" values are encoded in the URIs.

Any invalid URI entries received in the uri-data field are ignored by the client. If `OPTION_V6_SZTP_REDIRECT` does not contain at least one valid URI entry in the uri-data field, then the client MUST discard the option.

#### DHCPv6 Server Behavior

Section 18.3 of [RFC8415] governs server operation in regard to option assignment. As a convenience to the reader, we mention here that the server will send a particular option code only if configured with specific values for that option code and if the client requested it.

Option `OPTION_V6_SZTP_REDIRECT` is a singleton. Servers MUST NOT send more than one instance of the `OPTION_V6_SZTP_REDIRECT` option.

### 8.3. Common Field Encoding

Both of the DHCPv4 and DHCPv6 options defined in this section encode a list of bootstrap server URIs. The "URI" structure is a DHCP option that can contain multiple URIs (see [RFC7227], Section 5.7). Each URI entry in the bootstrap-server-list is structured as follows:

```

+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+...+---+---+---+---+
|      uri-length      |      URI      |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+...+---+---+---+---+

```

- \* uri-length: 2 octets long, specifies the length of the URI data.
- \* URI: URI of SZTP bootstrap server.

The URI of the SZTP bootstrap server MUST use the "https" URI scheme defined in Section 2.7.2 of [RFC7230], and MUST be in form "https://<ip-address-or-hostname>[:<port>]".



## 9. Security Considerations

### 9.1. Clock Sensitivity

The solution in this document relies on TLS certificates, owner certificates, and ownership vouchers, all of which require an accurate clock in order to be processed correctly (e.g., to test validity dates and revocation status). Implementations SHOULD ensure devices have an accurate clock when shipped from manufacturing facilities, and take steps to prevent clock tampering.

If it is not possible to ensure clock accuracy, it is RECOMMENDED that implementations disable the aspects of the solution having clock sensitivity. In particular, such implementations should assume that TLS certificates, ownership vouchers, and owner certificates never expire and are not revokable. From an ownership voucher perspective, manufacturers SHOULD issue a single ownership voucher for the lifetime of such devices.

Implementations SHOULD NOT rely on NTP for time, as NTP is not a secure protocol at this time. Note, there is an IETF work-in-progress to secure NTP [I-D.ietf-ntp-using-nts-for-ntp].

### 9.2. Use of IDevID Certificates

IDevID certificates, as defined in [Std-802.1AR-2018], are RECOMMENDED, both for the TLS-level client certificate used by devices when connecting to a bootstrap server, as well as for the device identity certificate used by owners when encrypting the SZTP bootstrapping data artifacts.

### 9.3. Immutable Storage for Trust Anchors

Devices MUST ensure that all their trust anchor certificates, including those for connecting to bootstrap servers and verifying ownership vouchers, are protected from external modification.

It may be necessary to update these certificates over time (e.g., the manufacturer wants to delegate trust to a new CA). It is therefore expected that devices MAY update these trust anchors when needed through a verifiable process, such as a software upgrade using signed software images.

### 9.4. Secure Storage for Long-lived Private Keys

Manufacturer-generated device identifiers may have very long lifetimes. For instance, [Std-802.1AR-2018] recommends using the "notAfter" value 99991231235959Z in IDevID certificates. Given the

long-lived nature of these private keys, it is paramount that they are stored so as to resist discovery, such as in a secure cryptographic processor, such as a trusted platform module (TPM) chip.

#### 9.5. Blindly Authenticating a Bootstrap Server

This document allows a device to blindly authenticate a bootstrap server's TLS certificate. It does so to allow for cases where the redirect information may be obtained in an unsecured manner, which is desirable to support in some cases.

To compensate for this, this document requires that devices, when connected to an untrusted bootstrap server, assert that data downloaded from the server is signed.

#### 9.6. Disclosing Information to Untrusted Servers

This document allows devices to establish connections to untrusted bootstrap servers. However, since the bootstrap server is untrusted, it may be under the control of an adversary, and therefore devices SHOULD be cautious about the data they send to the bootstrap server in such cases.

Devices send different data to bootstrap servers at each of the protocol layers TCP, TLS, HTTP, and RESTCONF.

At the TCP protocol layer, devices may relay their IP address, subject to network translations. Disclosure of this information is not considered a security risk.

At the TLS protocol layer, devices may use a client certificate to identify and authenticate themselves to untrusted bootstrap servers. At a minimum, the client certificate must disclose the device's serial number, and may disclose additional information such as the device's manufacturer, hardware model, public key, etc. Knowledge of this information may provide an adversary with details needed to launch an attack. It is RECOMMENDED that secrecy of the network constituency is not relied on for security.

At the HTTP protocol layer, devices may use an HTTP authentication scheme to identify and authenticate themselves to untrusted bootstrap servers. At a minimum, the authentication scheme must disclose the device's serial number and, concerningly, may, depending on the authentication mechanism used, reveal a secret that is only supposed to be known to the device (e.g., a password). Devices SHOULD NOT use an HTTP authentication scheme (e.g., HTTP Basic) with an untrusted

bootstrap server that reveals a secret that is only supposed to be known to the device.

At the RESTCONF protocol layer, devices use the "get-bootstrapping-data" RPC, but not the "report-progress" RPC, when connected to an untrusted bootstrap server. The "get-bootstrapping-data" RPC allows additional input parameters to be passed to the bootstrap server (e.g., "os-name", "os-version", "hw-model"). It is RECOMMENDED that devices only pass the "signed-data-preferred" input parameter to an untrusted bootstrap server. While it is okay for a bootstrap server to immediately return signed onboarding information, it is RECOMMENDED that bootstrap servers instead promote the untrusted connection to a trusted connection, as described in Appendix B, thus enabling the device to use the "report-progress" RPC while processing the onboarding information.

#### 9.7. Sequencing Sources of Bootstrapping Data

For devices supporting more than one source for bootstrapping data, no particular sequencing order has to be observed for security reasons, as the solution for each source is considered equally secure. However, from a privacy perspective, it is RECOMMENDED that devices access local sources before accessing remote sources.

#### 9.8. Safety of Private Keys used for Trust

The solution presented in this document enables bootstrapping data to be trusted in two ways, either through transport level security or through the signing of artifacts.

When transport level security (i.e., a trusted bootstrap server) is used, the private key for the end-entity certificate must be online in order to establish the TLS connection.

When artifacts are signed, the signing key is required to be online only when the bootstrap server is returning a dynamically generated signed-data response. For instance, a bootstrap server, upon receiving the "signed-data-preferred" input parameter to the "get-bootstrapping-data" RPC, may dynamically generate a response that is signed.

Bootstrap server administrators are RECOMMENDED to follow best practice to protect the private key used for any online operation. For instance, use of a hardware security module (HSM) is RECOMMENDED. If an HSM is not used, frequent private key refreshes are RECOMMENDED, assuming all bootstrapping devices have an accurate clock (see Section 9.1).

For best security, it is RECOMMENDED that owners only provide bootstrapping data that has been signed, using a protected private key, and encrypted, using the device's public key from its secure device identity certificate.

#### 9.9. Increased Reliance on Manufacturers

The SZTP bootstrapping protocol presented in this document shifts some control of initial configuration away from the rightful owner of the device and towards the manufacturer and its delegates.

The manufacturer maintains the list of well-known bootstrap servers its devices will trust. By design, if no bootstrapping data is found via other methods first, the device will try to reach out to the well-known bootstrap servers. There is no mechanism to prevent this from occurring other than by using an external firewall to block such connections. Concerns related to trusted bootstrap servers are discussed in Section 9.10.

Similarly, the manufacturer maintains the list of voucher signing authorities its devices will trust. The voucher signing authorities issue the vouchers that enable a device to trust an owner's domain certificate. It is vital that manufacturers ensure the integrity of these voucher signing authorities, so as to avoid incorrect assignments.

Operators should be aware that this system assumes that they trust all the pre-configured bootstrap servers and voucher signing authorities designated by the manufacturers. While operators may use points in the network to block access to the well-known bootstrap servers, operators cannot prevent voucher signing authorities from generating vouchers for their devices.

#### 9.10. Concerns with Trusted Bootstrap Servers

Trusted bootstrap servers, whether well-known or discovered, have the potential to cause problems, such as the following.

- o A trusted bootstrap server that has been compromised may be modified to return unsigned data of any sort. For instance, a bootstrap server that is only suppose to return redirect information might be modified to return onboarding information. Similarly, a bootstrap server that is only supposed to return signed data, may be modified to return unsigned data. In both cases, the device will accept the response, unaware that it wasn't supposed to be any different. It is RECOMMENDED that maintainers of trusted bootstrap servers ensure that their systems are not easily compromised and, in case of compromise, have mechanisms in

place to detect and remediate the compromise as expediently as possible.

- o A trusted bootstrap server hosting either unsigned, or signed but not encrypted, data may disclose information to unwanted parties (e.g., an administrator of the bootstrap server). This is a privacy issue only, but could reveal information that might be used in a subsequent attack. Disclosure of redirect information has limited exposure (it is just a list of bootstrap servers), whereas disclosure of onboarding information could be highly revealing (e.g., network topology, firewall policies, etc.). It is RECOMMENDED that operators encrypt the bootstrapping data when its contents are considered sensitive, even to the point of hiding it from the administrators of the bootstrap server, which may be maintained by a 3rd-party.

#### 9.11. Validity Period for Conveyed Information

The conveyed information artifact does not specify a validity period. For instance, neither redirect information nor onboarding information enable "not-before" or "not-after" values to be specified, and neither artifact alone can be revoked.

For unsigned data provided by an untrusted source of bootstrapping data, it is not meaningful to discuss its validity period when the information itself has no authenticity and may have come from anywhere.

For unsigned data provided by a trusted source of bootstrapping data (i.e., a bootstrap server), the availability of the data is the only measure of it being current. Since the untrusted data comes from a trusted source, its current availability is meaningful and, since bootstrap servers use TLS, the contents of the exchange cannot be modified or replayed.

For signed data, whether provided by an untrusted or trusted source of bootstrapping data, the validity is constrained by the validity of the both the ownership voucher and owner certificate used to authenticate it.

The ownership voucher's validity is primarily constrained by the ownership voucher's "created-on" and "expires-on" nodes. While [RFC8366] recommends short-lived vouchers (see Section 6.1), the "expires-on" node may be set to any point in the future, or omitted altogether to indicate that the voucher never expires. The ownership voucher's validity is secondarily constrained by the manufacturer's PKI used to sign the voucher; whilst an ownership voucher cannot be revoked directly, the PKI used to sign it may be.

The owner certificate's validity is primarily constrained by the X.509's validity field, the "notBefore" and "notAfter" values, as specified by the certificate authority that signed it. The owner certificate's validity is secondarily constrained by the validity of the PKI used to sign the voucher. Owner certificates may be revoked directly.

For owners that wish to have maximum flexibility in their ability to specify and constrain the validity of signed data, it is RECOMMENDED that a unique owner certificate is created for each signed artifact. Not only does this enable a validity period to be specified, for each artifact, but it also enables to the validity of each artifact to be revoked.

#### 9.12. Cascading Trust via Redirects

Redirect Information (Section 2.1), by design, instructs a bootstrapping device to initiate a HTTPS connection to the specified bootstrap servers.

When the redirect information is trusted, the redirect information can encode a trust anchor certificate used by the device to authenticate the TLS end-entity certificate presented by each bootstrap server.

As a result, any compromise in an interaction providing redirect information may result in compromise of all subsequent interactions.

#### 9.13. Possible Reuse of Private Keys

This document describes two uses for secure device identity certificates.

The primary use is for when the device authenticates itself to a bootstrap server, using its private key for TLS-level client-certificate based authentication.

A secondary use is for when the device needs to decrypt provided bootstrapping artifacts, using its private key to decrypt the data or, more precisely, per Section 6 in [RFC5652], decrypt a symmetric key used to decrypt the data.

This document, in Section 3.4 allows for the possibility that the same secure device identity certificate is used for both uses, as [Std-802.1AR-2018] states that a DevID certificate MAY have the "keyEncipherment" KeyUsage bit, in addition to the "digitalSignature" KeyUsage bit, set.

While it is understood that it is generally frowned upon to reuse private keys, this document views such reuse acceptable as there are not any known ways to cause a signature made in one context to be (mis)interpreted as valid in the other context.

#### 9.14. Non-Issue with Encrypting Signed Artifacts

This document specifies the encryption of signed objects, as opposed to the signing of encrypted objects, as might be expected given well-publicized oracle attacks (e.g., the padding oracle attack).

This document does not view such attacks as feasible in the context of the solution because the decrypted text never leaves the device.

#### 9.15. The "ietf-sztp-conveyed-info" YANG Module

The ietf-sztp-conveyed-info module defined in this document defines a data structure that is always wrapped by a CMS structure. When accessed by a secure mechanism (e.g., protected by TLS), then the CMS structure may be unsigned. However, when accessed by an insecure mechanism (e.g., removable storage device), then the CMS structure must be signed, in order for the device to trust it.

Implementations should be aware that signed bootstrapping data only protects the data from modification, and that the contents are still visible to others. This doesn't affect security so much as privacy. That the contents may be read by unintended parties when accessed by insecure mechanisms is considered next.

The ietf-sztp-conveyed-info module defines a top-level "choice" statement that declares the contents are either "redirect-information" or "onboarding-information". Each of these two cases are now considered.

When the content of the CMS structure is redirect-information, an observer can learn about the bootstrap servers the device is being directed to, their IP addresses or hostnames, ports, and trust anchor certificates. Knowledge of this information could provide an observer some insight into a network's inner structure.

When the content of the CMS structure is onboarding information, an observer could learn considerable information about how the device is to be provisioned. This information includes the operating system version, initial configuration, and script contents. This information should be considered sensitive and precautions should be taken to protect it (e.g., encrypt the artifact using the device's public key).

#### 9.16. The "ietf-sztp-bootstrap-server" YANG Module

The ietf-sztp-bootstrap-server module defined in this document specifies an API for a RESTCONF [RFC8040]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The NETCONF Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular users to a preconfigured subset of all available protocol operations and content.

This module presents no data nodes (only RPCs). There is no need to discuss the sensitivity of data nodes.

This module defines two RPC operations that may be considered sensitive in some network environments. These are the operations and their sensitivity/vulnerability:

**get-bootstrapping-data:** This RPC is used by devices to obtain their bootstrapping data. By design, each device, as identified by its authentication credentials (e.g. client certificate), can only obtain its own data. NACM is not needed to further constrain access to this RPC.

**report-progress:** This RPC is used by devices to report their bootstrapping progress. By design, each device, as identified by its authentication credentials (e.g. client certificate), can only report data for itself. NACM is not needed to further constrain access to this RPC.

### 10. IANA Considerations

#### 10.1. The IETF XML Registry

This document registers two URIs in the "ns" subregistry of the IETF XML Registry [RFC3688] maintained at <https://www.iana.org/assignments/xml-registry/xml-registry.xhtml#ns>. Following the format in [RFC3688], the following registrations are requested:

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



## 10.2. The YANG Module Names Registry

This document registers two YANG modules in the YANG Module Names registry [RFC6020] maintained at <https://www.iana.org/assignments/yang-parameters/yang-parameters.xhtml>. Following the format defined in [RFC6020], the below registrations are requested:

```

name:      ietf-sztp-conveyed-info
namespace: urn:ietf:params:xml:ns:yang:ietf-sztp-conveyed-info
prefix:    sztp-info
reference:  RFC XXXX

name:      ietf-sztp-bootstrap-server
namespace: urn:ietf:params:xml:ns:yang:ietf-sztp-bootstrap-server
prefix:    sztp-svr
reference:  RFC XXXX

```

## 10.3. The SMI Security for S/MIME CMS Content Type Registry

This document registers two SMI security codes in the "SMI Security for S/MIME CMS Content Type" registry (1.2.840.113549.1.9.16.1) maintained at <https://www.iana.org/assignments/smi-numbers/smi-numbers.xhtml#security-smime-1>. Following the format used in Section 3.4 of [RFC7107], the below registrations are requested:

Decimal	Description	References
-----	-----	-----
TBD1	id-ct-sztpConveyedInfoXML	[RFCXXXX]
TBD2	id-ct-sztpConveyedInfoJSON	[RFCXXXX]

id-ct-sztpConveyedInfoXML indicates that the "conveyed-information" is encoded using XML. id-ct-sztpConveyedInfoJSON indicates that the "conveyed-information" is encoded using JSON.

## 10.4. The BOOTP Manufacturer Extensions and DHCP Options Registry

This document registers one DHCP code point in the "BOOTP Manufacturer Extensions and DHCP Options" registry maintained at <http://www.iana.org/assignments/bootp-dhcp-parameters>. Following the format used by other registrations, the below registration is requested:

```

Tag:      143
Name:     OPTION_V4_SZTP_REDIRECT
Data Length: N
Meaning:  This option provides a list of URIs
           for SZTP bootstrap servers
Reference: [RFCXXXX]

```

Note: this request is to make permanent a previously registered early code point allocation.

#### 10.5. The Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Registry

This document registers one DHCP code point in "Option Codes" subregistry of the "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)" registry maintained at <http://www.iana.org/assignments/dhcpv6-parameters>. Following the format used by other registrations, the below registration is requested:

Value:	136
Description:	OPTION_V6_SZTP_REDIRECT
Client ORO:	Yes
Singleton Option:	Yes
Reference:	[RFCXXXX]

Note: this request is to make permanent a previously registered early code point allocation.

#### 10.6. The Service Name and Transport Protocol Port Number Registry

This document registers one service name in the Service Name and Transport Protocol Port Number Registry [RFC6335] maintained at <https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml>. Following the format defined in Section 8.1.1 of [RFC6335], the below registration is requested:

Service Name:	sztp
Transport Protocol(s):	TCP
Assignee:	IESG <iesg@ietf.org>
Contact:	IETF Chair <chair@ietf.org>
Description:	This service name is used to construct the SRV service label "_sztp" for discovering SZTP bootstrap servers.
Reference:	[RFCXXXX]
Port Number:	N/A
Service Code:	N/A
Known Unauthorized Uses:	N/A
Assignment Notes:	This protocol uses HTTPS as a substrate.

#### 10.7. The DNS Underscore Global Scoped Entry Registry

This document registers one service name in the DNS Underscore Global Scoped Entry Registry [I-D.ietf-dnsop-attrleaf] maintained at TBD\_IANA\_URL. Following the format defined in Section 4.3 of [I-D.ietf-dnsop-attrleaf], the below registration is requested:

RR Type:                   TXT  
\_NODE NAME:               \_sztp  
Reference:                 [RFCXXXX]

## 11. References

### 11.1. Normative References

- [I-D.ietf-dnsop-attrleaf]  
Crocker, D., "DNS Scoped Data Through "Underscore" Naming of Attribute Leaves", draft-ietf-dnsop-attrleaf-16 (work in progress), November 2018.
- [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/>>.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<https://www.rfc-editor.org/info/rfc1035>>.
- [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>>.
- [RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, DOI 10.17487/RFC2782, February 2000, <<https://www.rfc-editor.org/info/rfc2782>>.
- [RFC3396] Lemon, T. and S. Cheshire, "Encoding Long Options in the Dynamic Host Configuration Protocol (DHCPv4)", RFC 3396, DOI 10.17487/RFC3396, November 2002, <<https://www.rfc-editor.org/info/rfc3396>>.
- [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>>.

- [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>>.
- [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>>.
- [RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <<https://www.rfc-editor.org/info/rfc6762>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [RFC7227] Hankins, D., Mrugalski, T., Siodelski, M., Jiang, S., and S. Krishnan, "Guidelines for Creating New DHCPv6 Options", BCP 187, RFC 7227, DOI 10.17487/RFC7227, May 2014, <<https://www.rfc-editor.org/info/rfc7227>>.
- [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>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.

- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8366] Watsen, K., Richardson, M., Pritikin, M., and T. Eckert, "A Voucher Artifact for Bootstrapping Protocols", RFC 8366, DOI 10.17487/RFC8366, May 2018, <<https://www.rfc-editor.org/info/rfc8366>>.
- [RFC8415] Mrugalski, T., Siodelski, M., Volz, B., Yourtchenko, A., Richardson, M., Jiang, S., Lemon, T., and T. Winters, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 8415, DOI 10.17487/RFC8415, November 2018, <<https://www.rfc-editor.org/info/rfc8415>>.
- [Std-802.1AR-2018]  
IEEE SA-Standards Board, "IEEE Standard for Local and metropolitan area networks - Secure Device Identity", June 2018, <[https://standards.ieee.org/standard/802\\_1AR-2018.html](https://standards.ieee.org/standard/802_1AR-2018.html)>.

## 11.2. Informative References

- [I-D.ietf-netconf-crypto-types]  
Watsen, K. and H. Wang, "Common YANG Data Types for Cryptography", draft-ietf-netconf-crypto-types-02 (work in progress), October 2018.
- [I-D.ietf-netconf-trust-anchors]  
Watsen, K., "YANG Data Model for Global Trust Anchors", draft-ietf-netconf-trust-anchors-02 (work in progress), October 2018.
- [I-D.ietf-ntp-using-nts-for-ntp]  
Franke, D., Sibold, D., Teichel, K., Dansarie, M., and R. Sundblad, "Network Time Security for the Network Time Protocol", draft-ietf-ntp-using-nts-for-ntp-15 (work in progress), December 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>>.
- [RFC4250] Lehtinen, S. and C. Lonvick, Ed., "The Secure Shell (SSH) Protocol Assigned Numbers", RFC 4250, DOI 10.17487/RFC4250, January 2006, <<https://www.rfc-editor.org/info/rfc4250>>.

- [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>>.
- [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>>.
- [RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, DOI 10.17487/RFC6698, August 2012, <<https://www.rfc-editor.org/info/rfc6698>>.
- [RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <<https://www.rfc-editor.org/info/rfc6763>>.
- [RFC6891] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, DOI 10.17487/RFC6891, April 2013, <<https://www.rfc-editor.org/info/rfc6891>>.
- [RFC6960] Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 6960, DOI 10.17487/RFC6960, June 2013, <<https://www.rfc-editor.org/info/rfc6960>>.
- [RFC7107] Housley, R., "Object Identifier Registry for the S/MIME Mail Security Working Group", RFC 7107, DOI 10.17487/RFC7107, January 2014, <<https://www.rfc-editor.org/info/rfc7107>>.

- [RFC7766] Dickinson, J., Dickinson, S., Bellis, R., Mankin, A., and D. Wessels, "DNS Transport over TCP - Implementation Requirements", RFC 7766, DOI 10.17487/RFC7766, March 2016, <<https://www.rfc-editor.org/info/rfc7766>>.
- [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>>.
- [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. Example Device Data Model

This section defines a non-normative data model that enables the configuration of SZTP bootstrapping and discovery of what parameters are used by a device's bootstrapping logic.

### A.1. Data Model Overview

The following tree diagram provides an overview for the SZTP device data model.

```
module: example-device-data-model
  +--rw sztp
    +--rw enabled?                               boolean
    +--ro idevid-certificate?                     ct:end-entity-cert-cms
    | {bootstrap-servers}?
    +--ro bootstrap-servers {bootstrap-servers}?
    |   +--ro bootstrap-server* [address]
    |   |   +--ro address      inet:host
    |   |   +--ro port?       inet:port-number
    |   +--ro bootstrap-server-trust-anchors {bootstrap-servers}?
    |   |   +--ro reference*   ta:pinned-certificates-ref
    |   +--ro voucher-trust-anchors {signed-data}?
    |       +--ro reference*   ta:pinned-certificates-ref
```

In the above diagram, notice that there is only one configurable node "enabled". The expectation is that this node would be set to "true" in device's factory default configuration and that it would either be set to "false" or deleted when the SZTP bootstrapping is longer needed.

### A.2. Example Usage

Following is an instance example for this data model.



```
<sztp xmlns="https://example.com/sztp-device-data-model">
  <enabled>true</enabled>
  <idevid-certificate>base64encodedvalue==</idevid-certificate>
  <bootstrap-servers>
    <bootstrap-server>
      <address>sztp1.example.com</address>
      <port>8443</port>
    </bootstrap-server>
    <bootstrap-server>
      <address>sztp2.example.com</address>
      <port>8443</port>
    </bootstrap-server>
    <bootstrap-server>
      <address>sztp3.example.com</address>
      <port>8443</port>
    </bootstrap-server>
  </bootstrap-servers>
  <bootstrap-server-trust-anchors>
    <reference>manufacturers-root-ca-certs</reference>
  </bootstrap-server-trust-anchors>
  <voucher-trust-anchors>
    <reference>manufacturers-root-ca-certs</reference>
  </voucher-trust-anchors>
</sztp>
```

### A.3. YANG Module

The device model is defined by the YANG module defined in this section.

This module uses data types defined in [RFC6991], [I-D.ietf-netconf-crypto-types], and [I-D.ietf-netconf-trust-anchors].

```
module example-device-data-model {
  yang-version 1.1;
  namespace "https://example.com/sztp-device-data-model";
  prefix sztp-ddm;

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

  import ietf-crypto-types {
    prefix ct;
    revision-date 2018-06-04;
    description
```

```
    "This revision is defined in the -00 version of
      draft-ietf-netconf-crypto-types";
  reference
    "draft-ietf-netconf-crypto-types:
      Common YANG Data Types for Cryptography";
}

import ietf-trust-anchors {
  prefix ta;
  revision-date 2018-06-04;
  description
    "This revision is defined in -00 version of
      draft-ietf-netconf-trust-anchors.";
  reference
    "draft-ietf-netconf-trust-anchors:
      YANG Data Model for Global Trust Anchors";
}

organization
  "Example Corporation";

contact
  "Author: Bootstrap Admin <mailto:admin@example.com>";

description
  "This module defines a data model to enable SZTP
    bootstrapping and discover what parameters are used.
    This module assumes the use of an IDevID certificate,
    as opposed to any other client certificate, or the
    use of an HTTP-based client authentication scheme.";

revision 2019-01-15 {
  description
    "Initial version";
  reference
    "RFC XXXX: Secure Zero Touch Provisioning (SZTP)";
}

// features

feature bootstrap-servers {
  description
    "The device supports bootstrapping off bootstrap servers.";
}

feature signed-data {
  description
    "The device supports bootstrapping off signed data.";
```

```
}

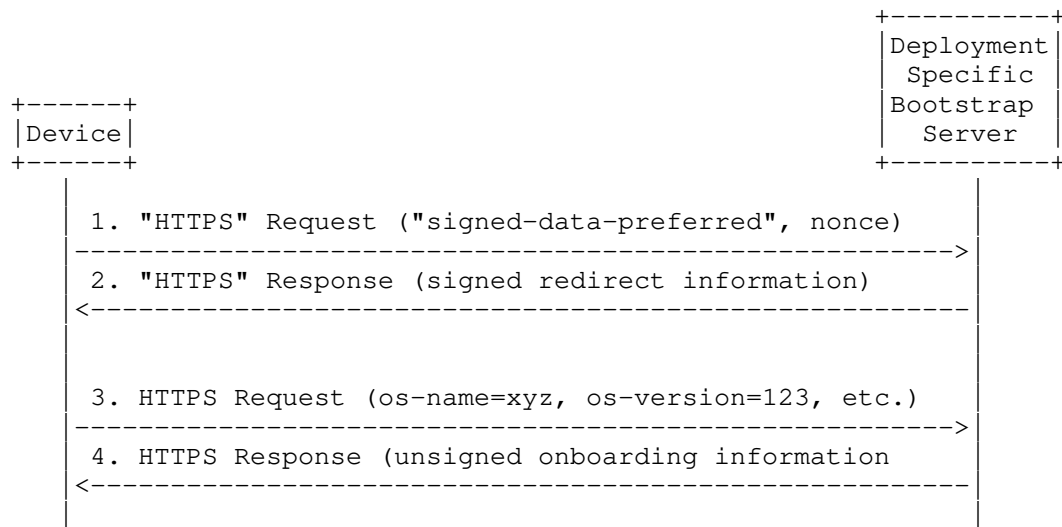
// protocol accessible nodes

container sztp {
  description
    "Top-level container for SZTP data model.";
  leaf enabled {
    type boolean;
    default false;
    description
      "The 'enabled' leaf controls if SZTP bootstrapping is
       enabled or disabled. The default is 'false' so that, when
       not enabled, which is most of the time, no configuration
       is needed.";
  }
  leaf idevid-certificate {
    if-feature bootstrap-servers;
    type ct:end-entity-cert-cms;
    config false;
    description
      "This CMS structure contains the IEEE 802.1AR-2009
       IDevID certificate itself, and all intermediate
       certificates leading up to, and optionally including,
       the manufacturer's well-known trust anchor certificate
       for IDevID certificates. The well-known trust anchor
       does not have to be a self-signed certificate.";
    reference
      "IEEE 802.1AR-2009:
       IEEE Standard for Local and metropolitan area
       networks - Secure Device Identity.";
  }
  container bootstrap-servers {
    if-feature bootstrap-servers;
    config false;
    description
      "List of bootstrap servers this device will attempt
       to reach out to when bootstrapping.";
    list bootstrap-server {
      key "address";
      description
        "A bootstrap server entry.";
      leaf address {
        type inet:host;
        mandatory true;
        description
          "The IP address or hostname of the bootstrap server the
           device should redirect to.";
      }
    }
  }
}
```

```
    }
    leaf port {
      type inet:port-number;
      default "443";
      description
        "The port number the bootstrap server listens on. If no
        port is specified, the IANA-assigned port for 'https'
        (443) is used.";
    }
  }
}
container bootstrap-server-trust-anchors {
  if-feature bootstrap-servers;
  config false;
  description "Container for a list of trust anchor references.";
  leaf-list reference {
    type ta:pinned-certificates-ref;
    description
      "A reference to a list of pinned certificate authority (CA)
      certificates that the device uses to validate bootstrap
      servers with.";
  }
}
container voucher-trust-anchors {
  if-feature signed-data;
  config false;
  description "Container for a list of trust anchor references.";
  leaf-list reference {
    type ta:pinned-certificates-ref;
    description
      "A reference to a list of pinned certificate authority (CA)
      certificates that the device uses to validate ownership
      vouchers with.";
  }
}
}
```

## Appendix B. Promoting a Connection from Untrusted to Trusted

The following diagram illustrates a sequence of bootstrapping activities that promote an untrusted connection to a bootstrap server to a trusted connection to the same bootstrap server. This enables a device to limit the amount of information it might disclose to an adversary hosting an untrusted bootstrap server.



The interactions in the above diagram are described below.

1. The device initiates an untrusted connection to a bootstrap server, as is indicated by putting "HTTPS" in double quotes above. It is still an HTTPS connection, but the device is unable to authenticate the bootstrap server's TLS certificate. Because the device is unable to trust the bootstrap server, it sends the "signed-data-preferred" input parameter, and optionally also the "nonce" input parameter, in the "get-bootstrapping-data" RPC. The "signed-data-preferred" parameter informs the bootstrap server that the device does not trust it and may be holding back some additional input parameters from the server (e.g., other input parameters, progress reports, etc.). The "nonce" input parameter enables the bootstrap server to dynamically obtain an ownership voucher from a MASA, which may be important for devices that do not have a reliable clock.
2. The bootstrap server, seeing the "signed-data-preferred" input parameter, knows that it can either send unsigned redirect information or signed data of any type. But, in this case, the bootstrap server has the ability to sign data and chooses to respond with signed redirect information, not signed onboarding information as might be expected, securely redirecting the device back to it again. Not displayed but, if the "nonce" input parameter was passed, the bootstrap server could dynamically connect to a download a voucher from the MASA having the nonce value in it. Details regarding a protocol enabling this integration is outside the scope of this document.

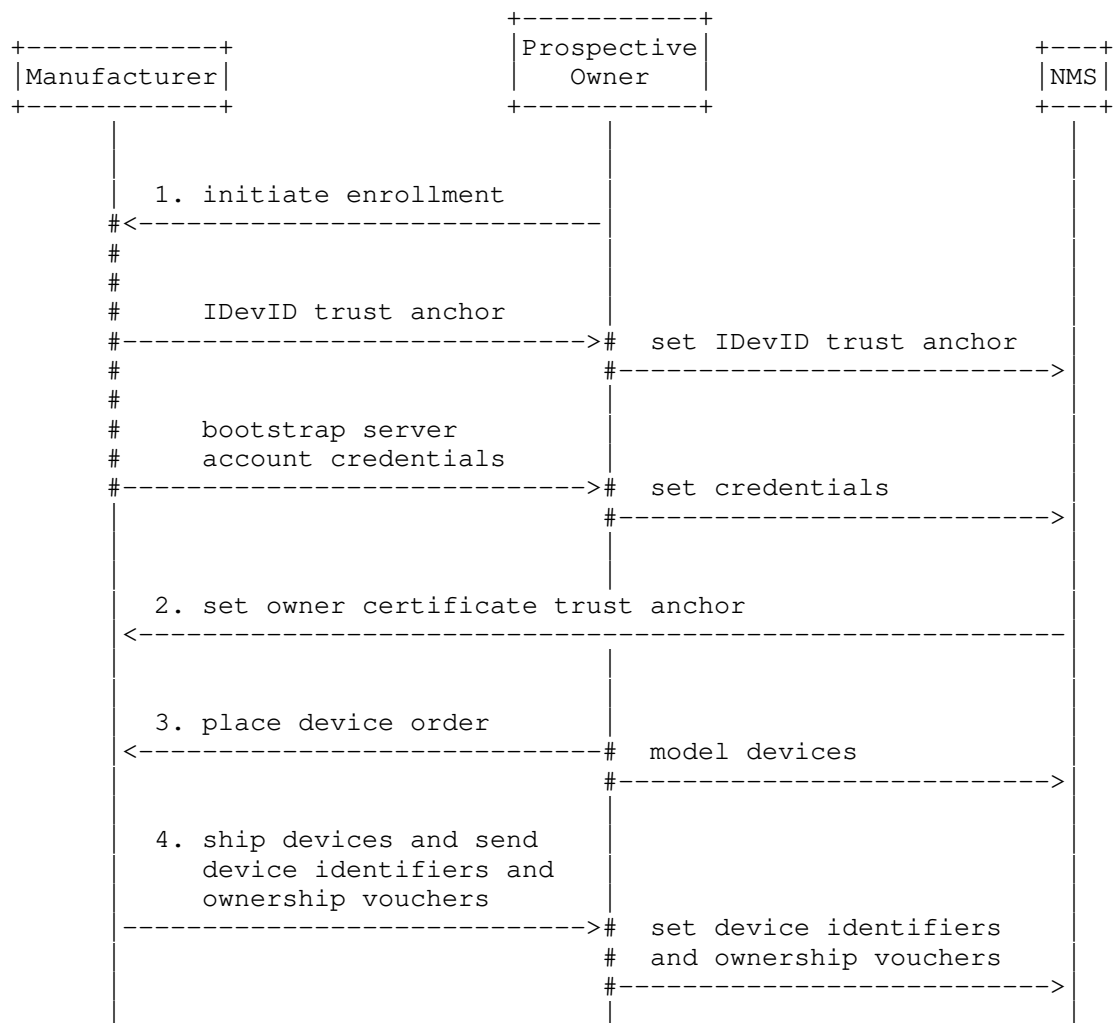
3. Upon validating the signed redirect information, the device establishes a secure connection to the bootstrap server. Unbeknownst to the device, it is the same bootstrap server it was connected to previously but, because the device is able to authenticate the bootstrap server this time, it sends its normal "get-bootstrapping-data" request (i.e., with additional input parameters) as well as its progress reports (not depicted).
4. This time, because the "signed-data-preferred" parameter was not passed, having access to all of the device's input parameters, the bootstrap server returns, in this example, unsigned onboarding information to the device. Note also that, because the bootstrap server is now trusted, the device will send progress reports to the server.

#### Appendix C. Workflow Overview

The solution presented in this document is conceptualized to be composed of the non-normative workflows described in this section. Implementation details are expected to vary. Each diagram is followed by a detailed description of the steps presented in the diagram, with further explanation on how implementations may vary.

##### C.1. Enrollment and Ordering Devices

The following diagram illustrates key interactions that may occur from when a prospective owner enrolls in a manufacturer's SZTP program to when the manufacturer ships devices for an order placed by the prospective owner.



Each numbered item below corresponds to a numbered item in the diagram above.

1. A prospective owner of a manufacturer's devices initiates an enrollment process with the manufacturer. This process includes the following:
  - \* Regardless how the prospective owner intends to bootstrap their devices, they will always obtain from the manufacturer the trust anchor certificate for the IDevID certificates. This certificate will be installed on the prospective owner's

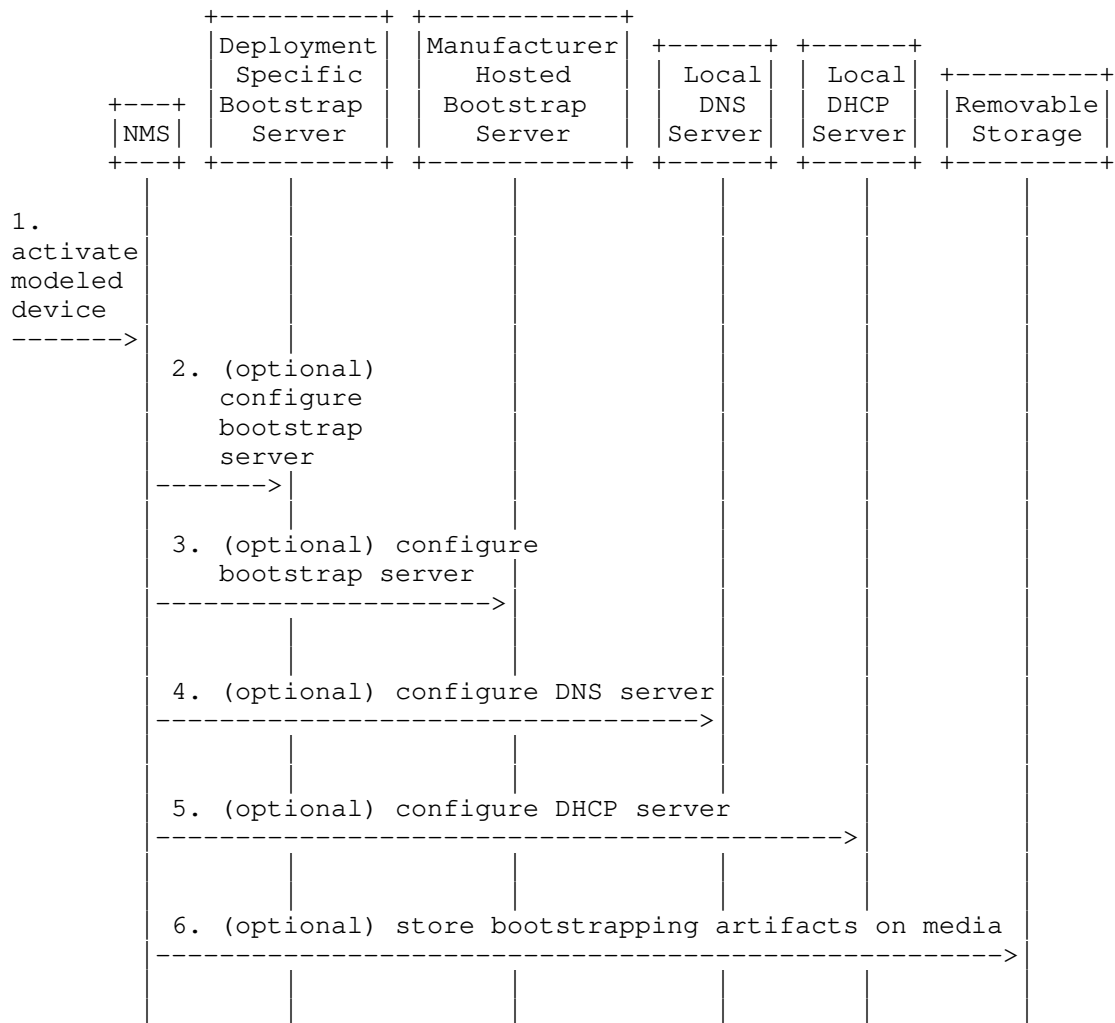
NMS so that the NMS can authenticate the IDevID certificates when they are presented to subsequent steps.

- \* If the manufacturer hosts an Internet based bootstrap server (e.g., a redirect server) such as described in Section 4.4, then credentials necessary to configure the bootstrap server would be provided to the prospective owner. If the bootstrap server is configurable through an API (outside the scope of this document), then the credentials might be installed on the prospective owner's NMS so that the NMS can subsequently configure the manufacturer-hosted bootstrap server directly.
2. If the manufacturer's devices are able to validate signed data (Section 5.4), and assuming that the prospective owner's NMS is able to prepare and sign the bootstrapping data itself, the prospective owner's NMS might set a trust anchor certificate onto the manufacturer's bootstrap server, using the credentials provided in the previous step. This certificate is the trust anchor certificate that the prospective owner would like the manufacturer to place into the ownership vouchers it generates, thereby enabling devices to trust the owner's owner certificate. How this trust anchor certificate is used to enable devices to validate signed bootstrapping data is described in Section 5.4.
  3. Some time later, the prospective owner places an order with the manufacturer, perhaps with a special flag checked for SZTP handling. At this time, or perhaps before placing the order, the owner may model the devices in their NMS, creating virtual objects for the devices with no real-world device associations. For instance the model can be used to simulate the device's location in the network and the configuration it should have when fully operational.
  4. When the manufacturer fulfills the order, shipping the devices to their intended locations, they may notify the owner of the devices' serial numbers and shipping destinations, which the owner may use to stage the network for when the devices power on. Additionally, the manufacturer may send one or more ownership vouchers, cryptographically assigning ownership of those devices to the owner. The owner may set this information on their NMS, perhaps binding specific modeled devices to the serial numbers and ownership vouchers.

#### C.2. Owner Stages the Network for Bootstrap

The following diagram illustrates how an owner might stage the network for bootstrapping devices.





Each numbered item below corresponds to a numbered item in the diagram above.

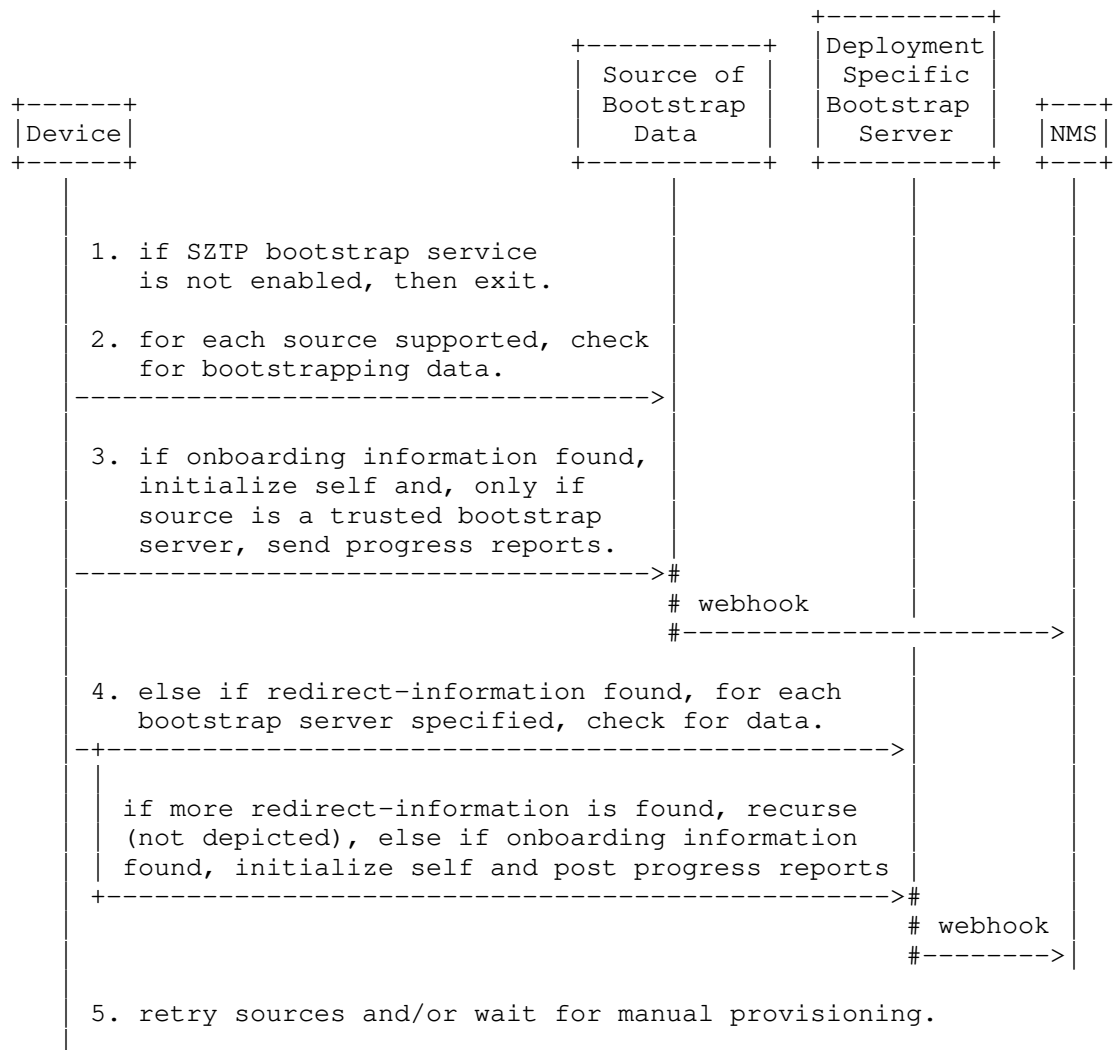
1. Having previously modeled the devices, including setting their fully operational configurations and associating device serial numbers and (optionally) ownership vouchers, the owner might "activate" one or more modeled devices. That is, the owner tells the NMS to perform the steps necessary to prepare for when the real-world devices power up and initiate the bootstrapping process. Note that, in some deployments, this step might be combined with the last step from the previous workflow. Here it

is depicted that an NMS performs the steps, but they may be performed manually or through some other mechanism.

2. If it is desired to use a deployment-specific bootstrap server, it must be configured to provide the bootstrapping data for the specific devices. Configuring the bootstrap server may occur via a programmatic API not defined by this document. Illustrated here as an external component, the bootstrap server may be implemented as an internal component of the NMS itself.
3. If it is desired to use a manufacturer hosted bootstrap server, it must be configured to provide the bootstrapping data for the specific devices. The configuration must be either redirect or onboarding information. That is, either the manufacturer hosted bootstrap server will redirect the device to another bootstrap server, or provide the device with the onboarding information itself. The types of bootstrapping data the manufacturer hosted bootstrap server supports may vary by implementation; some implementations may only support redirect information, or only support onboarding information, or support both redirect and onboarding information. Configuring the bootstrap server may occur via a programmatic API not defined by this document.
4. If it is desired to use a DNS server to supply bootstrapping data, a DNS server needs to be configured. If multicast DNS-SD is desired, then the DNS server must reside on the local network, otherwise the DNS server may reside on a remote network. Please see Section 4.2 for more information about how to configure DNS servers. Configuring the DNS server may occur via a programmatic API not defined by this document.
5. If it is desired to use a DHCP server to supply bootstrapping data, a DHCP server needs to be configured. The DHCP server may be accessed directly or via a DHCP relay. Please see Section 4.3 for more information about how to configure DHCP servers. Configuring the DHCP server may occur via a programmatic API not defined by this document.
6. If it is desired to use a removable storage device (e.g., USB flash drive) to supply bootstrapping data, the data would need to be placed onto it. Please see Section 4.1 for more information about how to configure a removable storage device.

### C.3. Device Powers On

The following diagram illustrates the sequence of activities that occur when a device powers on.



The interactions in the above diagram are described below.

1. Upon power being applied, the device checks to see if SZTP bootstrapping is configured, such as must be the case when running its "factory default" configuration. If SZTP bootstrapping is not configured, then the bootstrapping logic exits and none of the following interactions occur.
2. For each source of bootstrapping data the device supports, preferably in order of closeness to the device (e.g., removable

storage before Internet based servers), the device checks to see if there is any bootstrapping data for it there.

3. If onboarding information is found, the device initializes itself accordingly (e.g., installing a boot-image and committing an initial configuration). If the source is a bootstrap server, and the bootstrap server can be trusted (i.e., TLS-level authentication), the device also sends progress reports to the bootstrap server.

- \* The contents of the initial configuration should configure an administrator account on the device (e.g., username, SSH public key, etc.), and should configure the device either to listen for NETCONF or RESTCONF connections or to initiate call home connections [RFC8071], and should disable the SZTP bootstrapping service (e.g., the "enabled" leaf in data model presented in Appendix A).

- \* If the bootstrap server supports forwarding device progress reports to external systems (e.g., via a webhook), a "bootstrap-complete" progress report (Section 7.3) informs the external system to know when it can, for instance, initiate a connection to the device. To support this scenario further, the "bootstrap-complete" progress report may also relay the device's SSH host keys and/or TLS certificates, with which the external system can use to authenticate subsequent connections to the device.

If the device successfully completes the bootstrapping process, it exits the bootstrapping logic without considering any additional sources of bootstrapping data.

4. Otherwise, if redirect information is found, the device iterates through the list of specified bootstrap servers, checking to see if the bootstrap server has bootstrapping data for the device. If the bootstrap server returns more redirect information, then the device processes it recursively. Otherwise, if the bootstrap server returns onboarding information, the device processes it following the description provided in (3) above.
5. After having tried all supported sources of bootstrapping data, the device may retry again all the sources and/or provide manageability interfaces for manual configuration (e.g., CLI, HTTP, NETCONF, etc.). If manual configuration is allowed, and such configuration is provided, the configuration should also disable the SZTP bootstrapping service, as the need for bootstrapping would no longer be present.

## Appendix D. Change Log

## D.1. ID to 00

- o Major structural update; the essence is the same. Most every section was rewritten to some degree.
- o Added a Use Cases section
- o Added diagrams for "Actors and Roles" and "NMS Precondition" sections, and greatly improved the "Device Boot Sequence" diagram
- o Removed support for physical presence or any ability for configlets to not be signed.
- o Defined the Conveyed Information DHCP option
- o Added an ability for devices to also download images from configuration servers
- o Added an ability for configlets to be encrypted
- o Now configuration servers only have to support HTTP/S - no other schemes possible

## D.2. 00 to 01

- o Added boot-image and validate-owner annotations to the "Actors and Roles" diagram.
- o Fixed 2nd paragraph in section 7.1 to reflect current use of anyxml.
- o Added encrypted and signed-encrypted examples
- o Replaced YANG module with XSD schema
- o Added IANA request for the Conveyed Information DHCP Option
- o Added IANA request for media types for boot-image and configuration

## D.3. 01 to 02

- o Replaced the need for a configuration signer with the ability for each NMS to be able to sign its own configurations, using manufacturer signed ownership vouchers and owner certificates.

- o Renamed configuration server to bootstrap server, a more representative name given the information devices download from it.
- o Replaced the concept of a configlet by defining a southbound interface for the bootstrap server using YANG.
- o Removed the IANA request for the boot-image and configuration media types

#### D.4. 02 to 03

- o Minor update, mostly just to add an Editor's Note to show how this draft might integrate with the draft-pritikin-anima-bootstrapping-keyinfra.

#### D.5. 03 to 04

- o Major update formally introducing unsigned data and support for Internet-based redirect servers.
- o Added many terms to Terminology section.
- o Added all new "Guiding Principles" section.
- o Added all new "Sources for Bootstrapping Data" section.
- o Rewrote the "Interactions" section and renamed it "Workflow Overview".

#### D.6. 04 to 05

- o Semi-major update, refactoring the document into more logical parts
- o Created new section for information types
- o Added support for DNS servers
- o Now allows provisional TLS connections
- o Bootstrapping data now supports scripts
- o Device Details section overhauled
- o Security Considerations expanded
- o Filled in enumerations for notification types

## D.7. 05 to 06

- o Minor update
- o Added many Normative and Informative references.
- o Added new section Other Considerations.

## D.8. 06 to 07

- o Minor update
- o Added an Editorial Note section for RFC Editor.
- o Updated the IANA Considerations section.

## D.9. 07 to 08

- o Minor update
- o Updated to reflect review from Michael Richardson.

## D.10. 08 to 09

- o Added in missing "Signature" artifact example.
- o Added recommendation for manufacturers to use interoperable formats and file naming conventions for removable storage devices.
- o Added configuration-handling leaf to guide if config should be merged, replaced, or processed like an edit-config/yang-patch document.
- o Added a pre-configuration script, in addition to the post-configuration script from -05 (issue #15).

## D.11. 09 to 10

- o Factored ownership voucher and voucher revocation to a separate document: draft-kwatsen-netconf-voucher. (issue #11)
- o Removed <configuration-handling> options "edit-config" and "yang-patch". (issue #12)
- o Defined how a signature over signed-data returned from a bootstrap server is processed. (issue #13)

- o Added recommendation for removable storage devices to use open/standard file systems when possible. (issue #14)
- o Replaced notifications "script-[warning/error]" with "[pre/post]-script-[warning/error]". (goes with issue #15)
- o switched owner-certificate to be encoded using the PKCS #7 format. (issue #16)
- o Replaced md5/sha1 with sha256 inside a choice statement, for future extensibility. (issue #17)
- o A ton of editorial changes, as I went thru the entire draft with a fine-toothed comb.

## D.12. 10 to 11

- o fixed yang validation issues found by IETFYANGPageCompilation. note: these issues were NOT found by pyang --ietf or by the submission-time validator...
- o fixed a typo in the yang module, someone the config false statement was removed.

## D.13. 11 to 12

- o fixed typo that prevented Appendix B from loading the examples correctly.
- o fixed more yang validation issues found by IETFYANGPageCompilation. note: again, these issues were NOT found by pyang --ietf or by the submission-time validator...
- o updated a few of the notification enumerations to be more consistent with the other enumerations (following the warning/error pattern).
- o updated the information-type artifact to state how it is encoded, matching the language that was in Appendix B.

## D.14. 12 to 13

- o defined a standalone artifact to encode the old information-type into a PKCS #7 structure.
- o standalone information artifact hardcodes JSON encoding (to match the voucher draft).



- o combined the information and signature PKCS #7 structures into a single PKCS #7 structure.
- o moved the certificate-revocations into the owner-certificate's PKCS #7 structure.
- o eliminated support for voucher-revocations, to reflect the voucher-draft's switch from revocations to renewals.

## D.15. 13 to 14

- o Renamed "bootstrap information" to "onboarding information".
- o Rewrote DHCP sections to address the packet-size limitation issue, as discussed in Chicago.
- o Added Ian as an author for his text-contributions to the DHCP sections.
- o Removed the Guiding Principles section.

## D.16. 14 to 15

- o Renamed action "notification" to "update-progress" and, likewise "notification-type" to "update-type".
- o Updated examples to use "base64encodedvalue==" for binary values.
- o Greatly simplified the "Artifact Groupings" section, and moved it as a subsection to the "Artifacts" section.
- o Moved the "Workflow Overview" section to the Appendix.
- o Renamed "bootstrap information" to "update information".
- o Removed "Other Considerations" section.
- o Tons of editorial updates.

## D.17. 15 to 16

- o tweaked language to refer to "initial state" rather than "factory default configuration", so as accommodate white-box scenarios.
- o added a paragraph to Intro regarding how the solution primarily regards physical machines, but could be extended to VMs by a future document.

- o added a pointer to the Workflow Overview section (recently moved to the Appendix) to the Intro.
- o added a note that, in order to simplify the verification process, the "Conveyed Information" PKCS #7 structure MUST also contain the signing X.509 certificate.
- o noted that the owner certificate's must either have no Key Usage or the Key Usage must set the "digitalSignature" bit.
- o noted that the owner certificate's subject and subjectAltName values are not constrained.
- o moved/consolidated some text from the Artifacts section down to the Device Details section.
- o tightened up some ambiguous language, for instance, by referring to specific leaf names in the Voucher artifact.
- o reverted a previously overzealous s/unique-id/serial-number/change.
- o modified language for when ZTP runs from when factory-default config is running to when ZTP is configured, which the factory-defaults should set .

D.18. 16 to 17

- o Added an example for how to promote an untrusted connection to a trusted connection.
- o Added a "query parameters" section defining some parameters enabling scenarios raised in last call.
- o Added a "Disclosing Information to Untrusted Servers" section to the Security Considerations.

D.19. 17 to 18

- o Added Security Considerations for each YANG module.
- o Reverted back to the device always sending its DevID cert.
- o Moved data tree to "get-bootstrapping-data" RPC.
- o Moved the "update-progress" action to a "report-progress" RPC.

- o Added an "signed-data-preferred" parameter to "get-bootstrapping-data" RPC.
- o Added the "ietf-zerotouch-device" module.
- o Lots of small updates.

#### D.20. 18 to 19

- o Fixed "must" expressions, by converting "choice" to a "list" of "image-verification", each of which now points to a base identity called "hash-algorithm". There's just one algorithm currently defined (sha-256). Wish there was a standard crypto module that could identify such identities.

#### D.21. 19 to 20

- o Now references I-D.ietf-netmod-yang-tree-diagrams.
- o Fixed tree-diagrams in Section 2 to always reflect current YANG (now they are now dynamically generated).
- o The "redirect-information" container's "trust-anchor" is now a CMS structure that can contain a chain of certificates, rather than a single certificate.
- o The "onboarding-information" container's support for image verification reworked to be extensible.
- o Added a reference to the "Device Details" section to the new example-device-data-model module.
- o Clarified that the device must always pass its IDevID certificate, even for untrusted bootstrap servers.
- o Fixed the description statement for the "script" typedef to refer to the [pre/post]-script-[warning/error] enums, rather than the legacy script-[warning/error] enums.
- o For the get-bootstrapping-data RPC's input, removed the "remote-id" and "circuit-id" fields, and added a "hw-model" field.
- o Improved DHCP error handling text.
- o Added MUST requirement for DHCPv6 client and server implementing [RFC3396] to handle URI lists longer than 255 octets.

- o Changed the "configuration" value in onboarding-information to be type "binary" instead of "anydata".
- o Moved everything from PKCS#7 to CMS (this shows up as a big change).
- o Added the early code point allocation assignments for the DHCP Options in the IANA Considerations section, and updated the RFC Editor note accordingly.
- o Added RFC Editor request to replace the assigned values for the CMS content types.
- o Relaxed auth requirements from device needing to always send IDevID cert to device needing to always send authentication credentials, as this better matches what RFC 8040 Section 2.5 says.
- o Moved normative module "ietf-zerotouch-device" to non-normative module "example-device-data-model".
- o Updated Title, Abstract, and Introduction per discussion on list.

## D.22. 20 to 21

- o Now any of the three artifact can be encrypted.
- o Fixed some line-too-long issues.

## D.23. 21 to 22

- o Removed specifics around how scripts indicate warnings or errors and how scripts emit output.
- o Moved the SZTP Device Data Model section to the Appendix.
- o Modified the YANG module in the SZTP Device Data Model section to reflect the latest trust-anchors and keystore drafts.
- o Modified types in other YANG modules to more closely emulate what is in draft-ietf-netconf-crypto-types.

## D.24. 22 to 23

- o Rewrote section 5.6 (processing onboarding information) to be clearer about error handling and retained state. Specifically:

- \* Clarified that a script, upon having an error, must gracefully exit, cleaning up any state that might hinder subsequent executions.
- \* Added ability for scripts to be executed again with a flag enabling them to clean up state from a previous execution.
- \* Clarified that the configuration commit is atomic.
- \* Clarified that any error encountered after committing the configuration (e.g., in the "post-configuration-script") must rollback the configuration to the previous configuration.
- \* Clarified that failure to successfully deliver the "bootstrap-initiated" and "bootstrap-complete" progress types must be treated as an error.
- \* Clarified that "return to bootstrapping sequence" is to be interpreted in the recursive context. Meaning that the device rolls-back one loop, rather than start over from scratch.
- o Changed how a device verifies a boot-image from just "MUST match one of the supplied fingerprints" to also allow for the verification to use an cryptographic signature embedded into the image itself.
- o Added more "progress-type" enums for visibility reasons, enabling more strongly-typed debug information to be sent to the bootstrap server.
- o Added Security Considerations based on early SecDir review.
- o Added recommendation for device to send warning if the initial config does not disable the bootstrapping process.

## D.25. 23 to 24

- o Follow-ups from SecDir and Shepherd.
- o Added "boot-image-complete" enumeration.

## D.26. 24 to 25

- o Removed remaining old "bootstrapping information" term usage.
- o Fixed DHCP Option length definition.
- o Added reference to RFC 6187.

## D.27. 25 to 26

- o Updated URI structure text (sec 8.3) and added norm. ref to RFC7230 reflecting Alexey Melnikov's comment.
- o Added IANA registration for the 'zerotouch' service, per IESG review from Adam Roach.
- o Clarified device's looping behavior and support for alternative provisioning mechanisms, per IESG review from Mirja Kuehlewind.
- o Updated "ietf-sztp-bootstrap-server:ssh-host-key" from leaf-list to list, per IESG review from Benjamin Kaduk.
- o Added option size text to DHCPv4 option size to address Suresh Krishnan's IESG review discuss point.
- o Updated RFC3315 to RFC8415 and associated section references.
- o Revamped the DNS Server section, after digging into Alexey Melnikov comment.
- o Fixed IETF terminology template section in both YANG modules.

## D.28. 26 to 27

- o Added Security Consideration for cascading trust via redirects.
- o Modified the get-bootstrapping-data RPC's "nonce" input parameter to being a minimum of 16-bytes (used to be 8-bytes).
- o Added Security Consideration regarding possible reuse of device's private key.
- o Added Security Consideration regarding use of sign-then-encrypt.
- o Renamed "Zero Touch"/"zerotouch" throughout. Now uses "SZTP" when referring to the draft/solution, and "conveyed" when referring to the bootstrapping artifact.
- o Added missing text for "encrypted unsigned conveyed information" case.
- o Renamed "untrusted-connection" input paramter to "signed-data-preferred"
- o Switch yd:yang-data back to rc:yang-data

- o Added a couple features to the bootstrap-server module.

#### D.29. 27 to 28

- o Modified DNS section to no longer reference DNS-SD (now just plain TXT and SRV lookups, via multicast or unicast).
- o Registers "\_sztp" in the DNS Underscore Global Scoped Entry Registry.
- o Updated 802.1AR reference to current spec version.

#### Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by last name): Michael Behringer, Dean Bogdanovic, Martin Bjorklund, Joe Clarke, Dave Crocker, Toerless Eckert, Stephen Farrell, Stephen Hanna, Wes Hardaker, David Harrington, Mirja Kuehlewind, Radek Krejci, Suresh Krishnan, Benjamin Kaduk, David Mandelberg, Alexey Melnikov, Russ Mundy, Reinaldo Penno, Randy Presuhn, Max Pritikin, Michael Richardson, Adam Roach, Phil Shafer, Juergen Schoenwaelder.

Special thanks goes to Steve Hanna, Russ Mundy, and Wes Hardaker for brainstorming the original solution during the IETF 87 meeting in Berlin.

#### Authors' Addresses

Kent Watsen  
Juniper Networks

EMail: kwatsen@juniper.net

Mikael Abrahamsson  
T-Systems

EMail: mikael.abrahamsson@t-systems.se

Ian Farrer  
Deutsche Telekom AG

EMail: ian.farrer@telekom.de

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: September 14, 2017

M. Jethanandani  
Cisco Systems, Inc  
March 13, 2017

Accounting in NETCONF and RESTCONF  
draft-mahesh-netconf-accounting-01

Abstract

This document defines an accounting record for NETCONF and RESTCONF.

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on September 14, 2017.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of



the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
1.1. Terminology . . . . .	2
2. Accounting Record . . . . .	3
3. Data Model Definitions . . . . .	3
3.1. Data Organization . . . . .	3
3.2. YANG Module . . . . .	4
4. IANA Considerations . . . . .	8
5. Security Considerations . . . . .	9
6. Acknowledgements . . . . .	9
7. Normative References . . . . .	9
Author's Address . . . . .	10

## 1. Introduction

NETCONF [RFC6241] and RESTCONF [RFC8040] protocol operations are authenticated and authorized as part of the Authentication, Authorization and Accounting (AAA) framework. An accounting record needs to be created as part of the same framework for each of these operations to satisfy the accounting part of AAA. Having an accounting record that is consistent across vendors allows for the operator to compare operations across devices from different vendors. This document defines such a record and a corresponding YANG data model (ietf-netconf-am.yang).

The rest of this document will use NETCONF to imply both NETCONF and RESTCONF, but where applicable will call out each protocol specifically.

### 1.1. Terminology

The following terms are defined in NETCONF [RFC6241] and are not redefined here:

- o client
- o server
- o session
- o user

## 2. Accounting Record

An accounting record for NETCONF consists of the following fields.

acct	date	sr	sess	tas	use	gro	pat	act	rul	sta
-	-	c-	ion-	k-	r	ups	h	ion	e	tus
code	time	ip	id	id						

where:

acct-code: START indicates a start of a new record, NONE a continuation, and STOP the end of the record.

date-time: The date and time when the operation was performed (UTC Timezone)

src-ip: The source IP address that was used to request the operation

session-id: The session-id in case of NETCONF and would be blank in case of RESTCONF

task-id: Used to track a accounting record in case it needs to split for uploading or storing. The id is a monotonically increasing number assigned by the server.

user: The NETCONF user that requested this operation.

groups: The group the user belongs to.

path: The path in the NACM rule on which the operations is being performed

action: The action in the NACM rule

rule: The rule in the NACM that was used to authorize the action.

status: Whether the operations was permitted or denied.

## 3. Data Model Definitions

### 3.1. Data Organization

The following diagram highlights the contents and structure of the Accounting YANG module.

```
module: ietf-netconf-am
  +--ro nam
    +--ro accounting-record* [task-id]
      +--ro task-id      uint32
      +--ro session-id?  nc:session-id-type
      +--ro acct-code    enumeration
      +--ro date-time    yang:date-and-time
      +--ro src-ip       inet:ip-address
      +--ro group        nacm:group-name-type
      +--ro user?        nacm:user-name-type
      +--ro path         nacm:node-instance-identifier
      +--ro action       nacm:access-operations-type
      +--ro rule?        string
      +--ro status?      nacm:action-type
```

### 3.2. YANG Module

The following YANG module specifies the normative NETCONF content that MUST be supported by the server.

The "ietf-netconf-am" YANG module imports typedefs from YANG-TYPES [RFC6991], from NETCONF [RFC6241] and from NACM [RFC6536].

<CODE BEGINS> file "ietf-netconf-am@2017-03-13.yang"

```
module ietf-netconf-am {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-netconf-am";
  prefix "nam";

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

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

  import ietf-netconf {
    prefix nc;
    //reference
    // "RFC 6241: NETCONF Protocol";
  }
}
```

```
import ietf-netconf-acm {
  prefix nacm;
  //reference
  //      "RFC 6536: NACM";
}

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

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

  WG Chair: Mehmet Ersue
              <mailto:mehmet.ersue@nsn.com>

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

  Editor:     Mahesh Jethanandani
              <mailto:mjethanandani@gmail.com>";

description
  "This module defines an accounting record for NETCONF operations
  performed on the server. If these operations are authorized
  using rules defined by NACM [RFC6536], then that information is
  also captured by this module.

  Copyright (c) 2014 IETF Trust and the persons identified as
  authors of the code. All rights reserved.

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

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

  revision "2017-03-13" {
    description
      "Initial version";
    reference
      "RFC XXXX: NETCONF and RESTCONF Accounting";
  }
```

```
/*
 * Data definition statements.
 */

container nam {
  config false;
  description
    "Parameters for NETCONF Accounting Model.";

  list accounting-record {
    key "task-id";
    description
      "A list of accounting records generated by the server";

    leaf task-id {
      type uint32;
      description
        "The task-id is a monotonically increasing number
        assigned by the server to capture a single
        transaction.";
    }

    leaf session-id {
      type nc:session-id-type;
      description
        "If this operation happened over NETCONF, this
        field captures the NETCONF session-id. In case
        of RESTCONF this field can be left blank.";
    }

    leaf acct-code {
      type enumeration {
        enum start {
          description
            "Start of an accounting record";
        }
        enum stop {
          description
            "Indicates the end of an accounting
            record";
        }
        enum none {
          description
            "Indicates a single payload or a
            continuation of an accounting record.";
        }
      }
    }
  }
  mandatory true;
}
```

```
description
    "Some of the AAA server place a limit on the size
    of the payload that can be transmitted at any
    particular time.

    This field indicates what constitutes a complete
    accounting record by setting up the boundaries. If
    the accounting record fits within the payload
    boundary the field should be set to none."
}

leaf date-time {
    type yang:date-and-time;
    mandatory true;
    description
        "The date and time when the operation was
        requested."
}

leaf src-ip {
    type inet:ip-address;
    mandatory true;
    description
        "The source IP address where the request was made
        from."
}

leaf group {
    type nacm:group-name-type;
    mandatory true;
    description
        "The name of the group that the user who requested
        the operation belongs to."
}

leaf user {
    type nacm:user-name-type;
    description
        "The user within the group that is requesting this
        operation."
}

leaf path {
    type nacm:node-instance-identifier;
    mandatory true;
    description
        "Data Node Instance Identifier associated with the
        data node that the request is being made on."
```

```

        Instance identifiers start with the top-level
        data node, and a complete identifier is required
        for this value.";
    }

    leaf action {
        type nacm:access-operations-type;
        mandatory true;
        description
            "The type of NETCONF operation being requested.";
    }

    leaf rule {
        type string {
            length "1..max";
        }
        description
            "The name assigned to the rule that was used to
            authorize the action, if authorization was
            enabled.";
    }

    leaf status {
        type nacm:action-type;
        description
            "Action taken by the server when the above
            mentioned rule matched, if authorization was
            enable.";
    }
}
}
}
```

<CODE ENDS>

#### 4. IANA Considerations

This document makes two requests of IANA.

The first request is to register one URI in "The IETF XML Registry". Following the format in The IETF XML Registry [RFC3688], the following needs to be registered.

URI: urn:ietf:params:xml:ns:yang:ietf-netconf-am

Registrant Contact: The IESG

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

The second request is to register one module in the "YANG Module Names" registry. Following the format in YANG [RFC7950], the following needs to be registered.

Name: ietf-netconf-am

Namespace: urn:ietf:params:xml:ns:yang:ietf-netconf-am

Prefix: nam

Reference: RFC XXXX

Note to RFC Editor - Please replace XXXX with the RFC id assigned to this draft.

## 5. Security Considerations

The YANG module defined in this memo is designed to be accessed using the NETCONF protocol. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH.

Most of the data nodes defined in this YANG module are readonly, i.e. config false, and are therefore not vulnerable in network environments. However, they might contain data that might be sensitive and should be protected with the right NACM [RFC6536] rules.

## 6. Acknowledgements

## 7. 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, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<http://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, <<http://www.rfc-editor.org/info/rfc6241>>.



- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <<http://www.rfc-editor.org/info/rfc6536>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<http://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, <<http://www.rfc-editor.org/info/rfc7950>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<http://www.rfc-editor.org/info/rfc8040>>.

## Author's Address

Mahesh Jethanandani  
Cisco Systems, Inc  
170 West Tasman Drive  
San Jose, CA 95070  
USA

Email: [mjethanandani@gmail.com](mailto:mjethanandani@gmail.com)

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: January 18, 2018

M. Jethanandani  
Cisco Systems, Inc  
July 17, 2017

Accounting in NETCONF and RESTCONF  
draft-mahesh-netconf-accounting-03

Abstract

This document defines an accounting record for NETCONF and RESTCONF.

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on January 18, 2018.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
1.1. Compatability with remote AAA servers . . . . .	2
1.2. Terminology . . . . .	3
2. Accounting Record . . . . .	3
3. Data Model Definitions . . . . .	5
3.1. Data Organization . . . . .	5
3.2. YANG Module . . . . .	5
4. IANA Considerations . . . . .	10
5. Security Considerations . . . . .	10
6. Acknowledgements . . . . .	11
7. References . . . . .	11
7.1. Normative References . . . . .	11
7.2. Informative References . . . . .	12
Appendix A. Accounting model examples . . . . .	12
A.1. <edit-config> Example for Accounting . . . . .	12
A.2. <get> Example for Accounting . . . . .	12
A.3. NACM rule for Accounting . . . . .	13
Author's Address . . . . .	14

## 1. Introduction

NETCONF [RFC6241] and RESTCONF [RFC8040] protocol operations are authenticated and authorized as part of the Authentication, Authorization and Accounting (AAA) framework. An accounting record is generated as part of the same framework for each of these operations to satisfy the accounting part of AAA, but there has been no effort to define such a record. Having an accounting record that is consistent across vendors allows for the operator to compare operations across devices from different vendors. This document defines such a record and a corresponding YANG data model (ietf-netconf-am.yang).

The rest of this document will use NETCONF to imply both NETCONF and RESTCONF, but where applicable will call out each protocol specifically.

### 1.1. Compatability with remote AAA servers

This document does not cover how the server interacts with remote AAA servers and any interaction is out of scope of this document. A particular implementation can make the records available as part of <get> request, send a notification every time a accounting record is

generated or use any existing protocol to update the remote AAA server.

## 1.2. Terminology

The following terms are defined in NETCONF [RFC6241] and are not redefined here:

- o client
- o <get>
- o notification
- o server
- o session
- o user

And the following terms are defined in NACM [RFC6536] and are not redefined here.

- o data-node
- o action
- o rule

## 2. Accounting Record

An accounting record for NETCONF consists of the following fields. Note, there is no accounting record for reading or notification of an accounting record.

message-id	session-id	src-ip	destination	user	groups	rules	data-node	value	action	status
------------	------------	--------	-------------	------	--------	-------	-----------	-------	--------	--------

where:

message-id: This is the id within a given NETCONF session assigned to each RPC. RESTCONF has no concept of a session, so this field would be left blank.

session-id: The session-id in case of NETCONF and would be blank in case of RESTCONF. If the accounting record needs to be fragmented for any reason, it is suggested that this field not be repeated in subsequent packets. Instead a combination of start and end record marker, and the message-id should be used to reassemble fragmented records.

src-ip: The source IP address that was used to request the operation. If the accounting record needs to be fragmented for any reason, it is suggested that this field not be repeated in subsequent packets. Instead a combination of start and end record marker, and the message-id should be used to reassemble fragmented records.

date-time: The date and time when the operation was performed (UTC Timezone). If the accounting record needs to be fragmented for any reason, it is suggested that this field not be repeated in subsequent packets. Instead a combination of start and end record marker, and the message-id should be used to reassemble fragmented records.

user: The NETCONF user that requested this operation. If the accounting record needs to be fragmented for any reason, it is suggested that this field not be repeated in subsequent packets. Instead a combination of start and end record marker, and the message-id should be used to reassemble fragmented records.

groups: The group the user belongs to. If the accounting record needs to be fragmented for any reason, it is suggested that this field not be repeated in subsequent packets. Instead a combination of start and end record marker, and the message-id should be used to reassemble fragmented records.

data-node: The data-node in the NACM [RFC6536] rule on which the operations is being performed

value: The value that was set for any of the attributes in the request

action: The action in the NACM [RFC6536] rule

rule: The rule in the NACM [RFC6536] that was matched to authorize the action.

status: Whether the operations was permitted or denied.

### 3. Data Model Definitions

The model uses the NACM extension statement of default-deny-all to protect accounting records. Explicit rules have to be defined to be enable access to the accounting records.

#### 3.1. Data Organization

The following diagram highlights the contents and structure of the Accounting YANG module. For information on annotations, please refer to YANG Tree Diagrams [I-D.ietf-netmod-yang-tree-diagrams].

```
module: ietf-netconf-am
  +--rw nam
    +--rw enable-nam?          boolean
    +--ro accounting-record* [session-id message-id]
      +--ro session-id        nc:session-id-type
      +--ro message-id        uint32
      +--ro date-time         yang:date-and-time
      +--ro src-ip            inet:ip-address
      +--ro group              nacm:group-name-type
      +--ro user?             nacm:user-name-type
      +--ro rule?             string
      +--ro data-node         nacm:node-instance-identifier
      +--ro value?
      +--ro action            nacm:access-operations-type
      +--ro status?          nacm:action-type
```

#### 3.2. YANG Module

The following YANG module specifies the normative NETCONF content that MUST be supported by the server.

The "ietf-netconf-am" YANG module imports typedefs from YANG-TYPES [RFC6991], from NETCONF [RFC6241] and from NACM [RFC6536].

<CODE BEGINS> file "ietf-netconf-am@2017-07-16.yang"

```
module ietf-netconf-am {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-netconf-am";
  prefix "nam";

  import ietf-inet-types {
    prefix inet;
  }

  import ietf-yang-types {
```

```
    prefix yang;
  }

  import ietf-netconf {
    prefix nc;
  }

  import ietf-netconf-acm {
    prefix nacm;
  }

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

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

    WG Chair: Mehmet Ersue
               <mailto:mehmet.ersue@nsn.com>

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

    Editor:     Mahesh Jethanandani
               <mailto:mjethanandani@gmail.com>";

  description
    "This module defines an accounting record for NETCONF operations
    performed on the server. If these operations are authorized
    using rules defined by NACM [RFC6536], then that information is
    also captured by this module.

    Copyright (c) 2014 IETF Trust and the persons identified as
    authors of the code. All rights reserved.

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

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

  revision "2017-07-16" {
```

```
    description
        "Initial version";
    reference
        "RFC XXXX: NETCONF and RESTCONF Accounting";
}

/*
 * Data definition statements.
 */

container nam {
    nacm:default-deny-all;

    description
        "Parameters for NETCONF Accounting Model.";

    leaf enable-nam {
        type boolean;
        default true;
        description
            "Enable or disable generation of NETCONF
            accounting records. If 'true', accounting
            records will be generated. If set to 'false'
            no accounting records will be generated.";
    }

    list accounting-record {
        key "session-id message-id";
        config false;
        description
            "A list of accounting records generated by the server";

        leaf session-id {
            type nc:session-id-type;
            description
                "If this operation happened over NETCONF, this
                field captures the NETCONF session-id. In case
                of RESTCONF this field can be left blank.";
        }

        leaf message-id {
            type uint32;
            description
                "Id that is assigned to each RPC within a given
                NETCONF session. Should be blank in case of
                RESTCONF.";
        }
    }
}
```



```
leaf date-time {
  type yang:date-and-time;
  mandatory true;
  description
    "The date and time when the operation was
    requested.";
}

leaf src-ip {
  type inet:ip-address;
  mandatory true;
  description
    "The source IP address where the request was made
    from.";
}

leaf group {
  type nacm:group-name-type;
  mandatory true;
  description
    "The name of the group that the user who requested
    the operation belongs to.";
}

leaf user {
  type nacm:user-name-type;
  description
    "The user within the group that is requesting this
    operation.";
}

leaf rule {
  type string {
    length "1..max";
  }
  description
    "The name assigned to the rule that was used to
    authorize the action, if authorization was
    enabled.";
}

leaf data-node {
  type nacm:node-instance-identifier;
  mandatory true;
  description
    "Data Node Instance Identifier associated with the
    data node that the request is being made on.
```

```
Instance identifiers start with the top-level
data node, and a complete identifier is required
for this value.";
}

anydata value {
  description
    "An optional field, it contains the value of any
    of the attribute that form the record.

    It could be as simple as the filter value
    'http' specified that the user requested as part
    of the authorization request such as in this
    example:

    <filter>
      <name>http</name>
    </filter>

    or it could be value being set for a ssh port
    in this example:

    <ssh>
      <port>2022</port>
    </ssh>";
}

leaf action {
  type nacm:access-operations-type;
  mandatory true;
  description
    "The type of NETCONF operation being requested.";
}

leaf status {
  type nacm:action-type;
  description
    "Action taken by the server when the above
    mentioned rule matched, if authorization was
    enable.";
}
}
}
```

<CODE ENDS>

#### 4. IANA Considerations

This document makes two requests of IANA.

The first request is to register one URI in "The IETF XML Registry". Following the format in The IETF XML Registry [RFC3688], the following needs to be registered.

URI: urn:ietf:params:xml:ns:yang:ietf-netconf-am

Registrant Contact: The IESG

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

The second request is to register one module in the "YANG Module Names" registry. Following the format in YANG [RFC7950], the following needs to be registered.

Name: ietf-netconf-am

Namespace: urn:ietf:params:xml:ns:yang:ietf-netconf-am

Prefix: nam

Reference: RFC XXXX

Note to RFC Editor - Please replace XXXX here and in the rest of the draft with the RFC id assigned to this draft.

#### 5. Security Considerations

The YANG module defined in this document is designed to be accessed via network management protocol 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 layers is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC5246].

The NETCONF Access Control Model (NACM) [RFC6536] provides the means to restrict access for particular NETCONF or RESTCONF users to a pre-configured subset of all available NETCONF or RESTCONF protocol operations and content.

Most of the data nodes defined in this YANG module are readonly, i.e. config false, and are therefore not vulnerable to manipulation in network environments. However, they might contain data that might be sensitive and should be protected with the right NACM [RFC6536] rules.

## 6. Acknowledgements

## 7. References

### 7.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<http://www.rfc-editor.org/info/rfc3688>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<http://www.rfc-editor.org/info/rfc5246>>.
- [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, <<http://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, <<http://www.rfc-editor.org/info/rfc6242>>.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <<http://www.rfc-editor.org/info/rfc6536>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<http://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, <<http://www.rfc-editor.org/info/rfc7950>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<http://www.rfc-editor.org/info/rfc8040>>.

## 7.2. Informative References

[I-D.ietf-netmod-yang-tree-diagrams]  
Bjorklund, M. and L. Berger, "YANG Tree Diagrams", draft-ietf-netmod-yang-tree-diagrams-01 (work in progress), June 2017.

## Appendix A. Accounting model examples

### A.1. <edit-config> Example for Accounting

This example demonstrates how the configuration could be updated to enable accounting. The attribute in the model that enables accounting is enable-nam.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
  message-id="4">
  <edit-config xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
    <target>
      <candidate/>
    </target>
    <test-option>test-then-set</test-option>
    <error-option>rollback-on-error</error-option>
    <config>
      <nam xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-am">
        <enable-nam>true</enable-nam>
      </nam>
    </config>
  </edit-config>
</rpc>
```

### A.2. <get> Example for Accounting

This example demonstrates what a <get> request and response might look like.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
  message-id="1">
  <get>
    <filter>
      <nam xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-am">
        <accounting-record/>
      </nam>
    </filter>
  </get>
</rpc>

<rpc-reply message-id="1"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <nam xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-am">
      <accounting-record>
        <session-id>101</session-id>
        <message-id>100</message-id>
        <date-time>2017-06-19T16:39:57-08:00</date-time>
        <src-ip>172.20.39.46</src-ip>
        <group>netconf-group</group>
        <user>netconf</user>
        <path xmlns:acme="http://example.com/ns/itf">
          /acme:interfaces/acme:interface
        </path>
        <value>
          <name>GigabitEthernet0/0/0/0</name>
          <admin-state>UP</admin-state>
        </value>
        <action>read</action>
        <rule>51</rule>
        <status>permit</status>
      </accounting-record>
    </nam>
  </data>
</rpc-reply>
```

### A.3. NACM rule for Accounting

This example demonstrates how NACM could be configured to permit access to accounting records. Note, the model has default-deny-all set to prevent access to accounting records by default, and expects explicit rules to be configured to permit access.

```
<nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
  <rule-list>
    <name>root</name>
    <group>root</group>

    <rule>
      <name>allow-nam</name>
      <path xmlns:n="urn:ietf:params:xml:ns:yang:ietf-netconf-am">
        /n:nam
      </path>
      <access-operations>read</access-operations>
      <action>permit</action>
      <comment>
        Allow the root group read access to the /nam data.
      </comment>
    </rule>
  </rule-list>
</nacm>
```

## Author's Address

Mahesh Jethanandani  
Cisco Systems, Inc  
170 West Tasman Drive  
San Jose, CA 95070  
USA

Email: [mjethanandani@gmail.com](mailto:mjethanandani@gmail.com)

NETMOD  
Internet-Draft  
Intended status: Standards Track  
Expires: August 25, 2017

E. Voit  
Cisco Systems  
A. Bierman  
YumaWorks  
A. Clemm  
Huawei  
T. Jenkins  
Cisco Systems  
February 21, 2017

YANG Notification Headers and Bundles  
draft-voit-netmod-yang-notifications2-00

Abstract

There are useful capabilities not available with existing YANG notifications as described in Section 7.16 of [RFC7950]. These include:

1. what are the set of transport agnostic header objects which might be usefully placed within YANG notifications.
2. how might a set of YANG notifications be bundled into a single transport message.
3. how do you query the originator of a notification to troubleshoot the bundling process

This specification provides technologies enabling these three capabilities.

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 <http://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 August 25, 2017.



## Copyright Notice

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

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

## Table of Contents

1. Introduction . . . . .	2
2. Terminology . . . . .	3
3. Header Objects . . . . .	3
4. Headers added to an RFC7950 Notification . . . . .	4
5. Bundled Notifications . . . . .	5
6. Querying an Object Model . . . . .	7
7. Data Model . . . . .	9
8. Security Considerations . . . . .	19
9. References . . . . .	20
9.1. Normative References . . . . .	20
9.2. Informative References . . . . .	20
Appendix A. Issues being worked . . . . .	20
Authors' Addresses . . . . .	21

## 1. Introduction

Mechanisms to support subscription to event notifications and yang datastore push are being defined in [sn] and [yang-push]. Work on those documents has shown that additional capabilities in YANG notifications would be helpful. Three of these capabilities include:

1. what are the set of transport agnostic header objects which might be usefully placed within YANG notifications.
2. how might a set of YANG notifications be bundled into a single transport message.
3. how do you query the originator of a notification to troubleshoot the bundling process.

As none of these three capabilities are specific to subscriptions, it would be good to define them in a transport protocol agnostic way.

## 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].

Definitions of Notification, Event, Event Notification, Publisher, Receiver, and Subscription are defined in [sn].

## 3. Header Objects

There are a number of transport independent headers which should have common definition across applications. These include:

- o record-type: what kind of information and have been assembled as part of this notification. (E.g., is it a YANG datastore update, an alarm, a syslog message, etc.)
- o subscription-id: provides a reference into the reason the originator believed the receiver wishes to be notified of this specific information.
- o record-severity: how important the originator feels this message to be.
- o record-time: the time an event notification itself was recorded in the originating system.
- o record-id: identifies an event notification on an originator.
- o observation-domain-id: identifies the originator process which discovered and recorded the event notification. (note: look to reuse the domains set up with IPFIX.)
- o notification-time: the time the message was packaged sent to the transport layer for delivery to the receiver.
- o signature: allows an application to sign a message so that a receiver can verify the authenticity of the message.
- o notification-id: identifies a message which includes one or more event records.

- o previous-notification-id: the Notification id previously sent to a receiver. When used in conjunction with notification-id, this allow loss/duplication across previous messages to be discovered.
- o message-generator-id: process which created the message notification. Allows identification of different line cards sending the notification messages. Used in conjunction with previous-notification-id, can help find drops and duplication when notifications are coming from multiple sources on a device. The logic is simple: if there is a message-generator-id in the header, then the previous-notification-id should been the notification-id the last time the message-generator-id was sent.

#### 4. Headers added to an RFC7950 Notification

With the headers defined, they may now be applied to extend an RFC-7950 notification. This section provides examples of this.

The first thing which is done is to encapsulate these header fields within their own subtree in the notification message so that these objects can easily be decoupled, processed, and removed from any notification record payload.

It is useful to sequence these objects so that processing by the receiver is as efficient as possible, allowing the discarding of uninteresting notifications as quickly as possible. One record priority encoding would include the objects presented in the sequence above to help minimize event record processing CPU cycles. (Need to add more here, and acknowledge that different payloads and systems might benefit from alternative sequencing.)

```
+---n notification-message
  +--ro notification-message-header
    | +--ro record-time
    | +--ro record-type?
    | +--ro record-id?
    | +--ro record-severity?
    | +--ro observation-domain-id?
    | +--ro subscription-id?
    | +--ro notification-time?
    | +--ro notification-id?
    | +--ro previous-notification-id?
    | +--ro signature?
    | +--ro message-generator-id?
  +--ro receiver-record-contents?
```

An actual instance of a notification might look like:

```
<notification
  xmlns="urn:ietf:params:xml:ns:netmod:notification:2.0">
  <notification-message-header>
    <record-time>
      2017-02-14T00:00:02Z
    </record-time>
    <record-type>
      yang-patch
    </record-type>
    <subscription-identifier>
      823472
    </subscription-identifier>
    <notification-time>
      2017-02-14T00:00:05Z
    </notification-time>
    <notification-identifier>
      456
    </notification-identifier>
    <previous-notification-identifier>
      567
    </previous-notification-identifier>
    <signature>
      lKIo8s03fd23.....
    </signature>
  </notification-message-header>
  <datastore-changes>
    ...(yang patch here)...
  </datastore-changes>
</notification>
```

## 5. Bundled Notifications

In many implementations, it may be inefficient to transport every notification independently. Instead, scale and processing speed can be improved by placing multiple notifications into one transportable bundle.

When this is done, one additional of a header field becomes valuable. This is the "record-count" which would tally the quantity of records which make up the contents of the bundle.

The format of a bundle would look as below. When compared to the unbundled notification, note that the headers have been split so that one set of headers associated with the notification occur once at the beginning of the message, and additional record specific headers which occur before individual records.

```
+---n bundled-notification-message
+--ro notification-message-header
|   +---ro notification-time
|   +---ro notification-id?
|   +---ro previous-notification-id?
|   +---ro signature?
|   +---ro message-generator-id?
|   +---ro record-count?
+--ro notification-records*
+--ro notification-record-header
|   +---ro record-time
|   +---ro record-type?
|   +---ro record-id?
|   +---ro record-severity?
|   +---ro observation-domain-id?
|   +---ro subscription-id?
+--ro receiver-record-contents?
```

An actual instance of a bundled notification might look like:

```

<notification
  xmlns="urn:ietf:params:xml:ns:netmod:notification:2.0">
  <bundled-notification-message-header>
    <notification-time>
      2017-02-14T00:00:05Z
    </notification-time>
    <notification-identifier>
      456
    </notification-identifier>
    <previous-notification-identifier>
      567
    </previous-notification-identifier>
    <signature>
      lKIo8s03fd23...
    </signature>
    <record-count>
      2
    </record-count>
  </bundled-notification-message-header>
  <notification-record>
    <notification-record-header>
      <record-time>
        2017-02-14T00:00:02Z
      </record-time>
      <record-type>
        yang-patch
      </record-type>
      <subscription-identifier>
        823472
      </subscription-identifier>
    </notification-record-header>
    <notification
      xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
      <datastore-changes>
        ...(yang patch here)...
      </datastore-changes>
    </notification>
  </notification-record>
  <notification-record>
    ...(record #2)...
  </notification-record>
</notification>

```

## 6. Querying an Object Model

It is possible that an administrator would like to examine the contents of notifications via random access using a YANG model. There could be several values in such random access. These include:

- o ability for applications to determine what message bundles were used to transport specific records.
- o ability for applications to check which receivers have been sent an event notification.
- o ability for applications to determine the time delta between event identification and transport.
- o ability to reconstruct message passing during troubleshooting.
- o ability to extract messages and records to evaluate whether the security filters have been properly applied.
- o ability to compare the payloads of the same notification message sent to different receivers (again to evaluate the impact of the security filtering).

If such random access is needed, the YANG model structure below would enable random access to the information.

```

+--ro notification-records
|   +--ro notification-record* [record-id]
|   |   +--ro record-time                yang:date-and-time
|   |   +--ro record-type                notification-record-format-type
|   |   +--ro record-id                  uint32
|   |   +--ro record-severity?           string
|   |   +--ro observation-domain-id?     string
|   |   +--ro notification-record-contents
|   |   +--ro subscription-id*           subscription-ref
+--ro notification-messages
|   +--ro notification-message* [notification-id]
|   |   +--ro notification-id            uint32
|   |   +--ro signature?                 string
|   |   +--ro message-generator-id?     string
|   |   +--ro notification-record        notification-record-ref
|   |   +--ro receiver-notification-messages
|   |   |   +--ro receiver-notification-message*
|   |   |   |   +--ro receiver?          receiver-ref
|   |   |   |   +--ro notification-time  yang:date-and-time
|   |   |   |   +--ro previous-notification-id? uint32
|   |   |   |   +--ro receiver-record-contents
|   |   +--ro bundled-notification-messages
|   |   |   +--ro bundled-notification-message* [notification-id]
|   |   |   |   +--ro notification-id    uint32
|   |   |   |   +--ro signature?         string
|   |   |   |   +--ro message-generator-id? string
|   |   |   |   +--ro included-notification-records
|   |   |   |   |   +--ro included-notification-record*
|   |   |   |   |   |   +--ro notification-record? notification-record-ref
|   |   |   +--ro receiver-notification-messages
|   |   |   |   +--ro receiver-notification-message*
|   |   |   |   |   +--ro receiver?      receiver-ref
|   |   |   |   |   +--ro notification-time yang:date-and-time
|   |   |   |   |   +--ro previous-notification-id? uint32
|   |   |   |   |   +--ro record-count?   uint16
|   |   |   |   +--ro included-notification-records
|   |   |   |   |   +--ro notification-record*
|   |   |   |   |   |   +--ro receiver-record-contents

```

If such random access is not seen as needed, the model above should be discarded. This will also simplify the YANG model in the following section.

## 7. Data Model

```

<CODE BEGINS> file "ietf-yang-notifications2.yang"
module ietf-yang-notifications2 {
  yang-version 1.1;

```



```
namespace "urn:ietf:params:xml:ns:yang:ietf-yang-notifications2";
prefix yn2;

import ietf-yang-types {
  prefix yang;
}
import ietf-subscribed-notifications {
  prefix sn;
}

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

  WG Chair:   Lou Berger
              <mailto:lberger@labn.net>

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

  Editor:     Eric Voit
              <mailto:evoit@cisco.com>

  Editor:     Alexander Clemm
              <mailto:ludwig@clemm.org>

  Editor:     Tim Jenkins
              <mailto:timjenki@cisco.com>

  Editor:     Andy Bierman
              <mailto:andy@yumaworks.com>" ;

description
  "This module contains conceptual YANG specifications for NETCONF
  Event Notifications.";

revision 2017-02-23 {
  description
    "This module includes several definitions for two new yang
    notification message formats:
    (a) a message format including the definitions for common header
        information prior to notification payload.
    (b) a message format allowing the bundling of multiple
        notifications within it

    It also includes data nodes for querying related information such
```

```
as:
  - ability to see contents of notifications both before and
    after any NACM filtering has been applied.
  - ability to see which message numbers have been sent to which
    receivers.";

reference
  "draft-voit-netmod-yang-notifications2-00";
}

/*
 * IDENTITIES
 */

/* Identities for notification record types */

identity notification-record-format {
  description
    "Base identity to represent a different formats for notification
    records.";
}

identity system-event {
  base notification-record-format;
  description
    "System XML event";
}

identity yang-datastore {
  base notification-record-format;
  description
    "yang tree info";
}

identity yang-patch {
  base notification-record-format;
  description
    "yang patch record";
}

identity syslog-entry {
  base notification-record-format;
  description
    "Entry into syslog (figure linkage to existing draft.");
}

identity alarm {
  base notification-record-format;
```

```
    description
      "Alarm (perhaps link draft-sharma-netmod-fault-model-01 for more
        info)";
  }

/*
 * TYPEDEFS
 */

typedef notification-record-ref {
  type leafref {
    path "/notification-records/notification-record/record-id";
  }
  description
    "This type is used to reference a notification record (a.k.a.,
      event).";
}

typedef receiver-ref {
  type leafref {
    path "/sn:subscriptions/sn:subscription/sn:receivers/"+
      "sn:receiver/sn:address";
  }
  description
    "This type is used to reference a receiver.";
}

typedef subscription-ref {
  type leafref {
    path "/sn:subscriptions/sn:subscription/sn:identifier";
  }
  description
    "This type is used to reference a receiver.";
}

typedef notification-record-format-type {
  type identityref {
    base notification-record-format;
  }
  description
    "Type of notification record";
}

/*
 * GROUPINGS
 */

grouping notification-message-header {
```

```
description
  "Header information included with a notification.";
leaf notification-id {
  type uint32;
  description
    "unique id for a notification which may go to one or many
    receivers.";
}
leaf signature {
  type string;
  description
    "Any originator signing of the contents of a notification
    message. This can be useful for originating applications to
    verify record contents even when shipping over unsecure
    transport.";
}
leaf message-generator-id {
  type string;
  description
    "Software entity which created the notification message (e.g.,
    linecard 1).";
}
}

grouping notification-message-receiver-header {
  description
    "Header information included with a notification which is
    specific to a receiver.";
  leaf notification-time {
    type yang:date-and-time;
    description
      "time the notification was generated prior to being sent to
      transport.";
  }
  leaf previous-notification-id {
    type uint32;
    description
      "Notification id previously sent by publisher to a specific
      receiver (allows detection of loss/duplication).";
  }
}

grouping notification-record-header {
  description
    "Common informational objects which might help a receiver
    interpret the meaning, details, and importance of an event
    notification.";
  leaf record-time {
```

```
    type yang:date-and-time;
    mandatory true;
    description
        "Time the system recognized the occurrence of an event.";
}
leaf record-type {
    type notification-record-format-type;
    description
        "Describes the type of payload included. This is turn allow
        the interpretation of the record contents.";
}
leaf record-id {
    type uint32;
    description
        "Identifier for the notification record.";
}
leaf record-severity {
    type string;
    description
        "System assigned severity. (Likely we need to build/find an
        enumeration of common ones.)";
}
leaf observation-domain-id {
    type string;
    description
        "Software entity which created the notification record (e.g.,
        process id).";
}
}

grouping subscribed-notification-record-header {
    description
        "Header information included with a notification.";
    uses notification-record-header;
    leaf subscription-id {
        type uint32;
        description
            "Id of the subscription which led to the notification being
            generated.";
    }
}

/*
 * NOTIFICATIONS
 */

notification notification-message {
    description
```

```
"Notification message to a receiver containing only one event.";
container notification-message-header {
  description
    "delineates header info from notification messages for easy
    parsing.";
  uses subscribed-notification-record-header;
  uses notification-message-header;
  uses notification-message-receiver-header;
}
anydata receiver-record-contents {
  description
    "Non-header info of what actually got sent to receiver after
    security filter. (Note: Possible to have extra process
    encryption.)";
}
}

notification bundled-notification-message {
  description
    "Notification message to a receiver containing many events,
    possibly relating to independent subscriptions.";
  container bundled-notification-message-header {
    description
      "Delineates header info from notification messages for easy
      parsing.";
    uses notification-message-header;
    uses notification-message-receiver-header {
      refine notification-time {
        mandatory true;
      }
    }
  }
  leaf record-count {
    type uint16;
    description
      "Quantity of events in a bundled-notification-message
      for a specific receiver. This value is per receiver in
      case an entire notification record is filtered out.";
  }
}

list notification-records {
  description
    "Set of messages within a notification to a receiver.";
  container notification-record-header {
    description
      "delineates header info from notification messages for easy
      parsing.";
    uses subscribed-notification-record-header;
  }
}
```

```
    }
    anydata receiver-record-contents {
      description
        "Non-header info of what actually got sent to receiver after
        security filter. (Note: Possible to have extra process
        encryption.)";
    }
  }
}

/*
 * DATA NODES
 */

container notification-records {
  config false;
  description
    "Maintains instances of event notifications recorded by the
    system.";
  list notification-record {
    key "record-id";
    description
      "Specific instances of event notifications recorded by the
      system.";
    uses notification-record-header {
      refine record-id {
        mandatory true;
      }
      refine record-type {
        mandatory true;
      }
    }
  }
  anydata notification-record-contents {
    mandatory true;
    description
      "Notification event contents independent of any receiver
      security filtering.";
  }
  leaf-list subscription-id {
    type subscription-ref;
    description
      "Instances of subscriptions which should receive or have
      received this event notification record.";
  }
}

}
container notification-messages {
```

```
config false;
description
  "Contains a history of the notification messages which have been
  generated.";
list notification-message {
  key "notification-id";
  description
    "Instances of notification messages generated with the intent
    of sending them to one or more receivers.";
  uses notification-message-header {
    refine notification-id {
      mandatory true;
    }
  }
}
leaf notification-record {
  type notification-record-ref;
  mandatory true;
  description
    "Included notification. The record itself, or elements of
    this record might not be sent to any included receiver based
    on security permissions for that receiver.";
}
container receiver-notification-messages {
  description
    "Contains a history of messages targeted for a receiver.";
  list receiver-notification-message {
    description
      "Maintains instances of messages targeted for a receiver.";
    leaf receiver {
      type receiver-ref;
      description
        "Reference to the recipient targeted for this
        notification message. (This also allows the unique
        identification of the subscription.)";
    }
    uses notification-message-receiver-header {
      refine notification-time {
        mandatory true;
      }
    }
    anydata receiver-record-contents {
      mandatory true;
      description
        "The specific security filtered contents of one record
        going to a receiver.";
    }
  }
}
```



```
    }  
  }  
  container bundled-notification-messages {  
    config false;  
    description  
      "Contains a history of bundled notification messages which have  
      been generated.";  
    list bundled-notification-message {  
      key "notification-id";  
      min-elements 1;  
      description  
        "Maintains instances of a bundled notification messages  
        generated with the intent of sending them to one or more  
        receivers.";  
      uses notification-message-header {  
        refine notification-id {  
          mandatory true;  
        }  
      }  
    }  
    container included-notification-records {  
      description  
        "Contains specific records included in the bundle.";  
      list included-notification-record {  
        description  
          "A specific instance of record included in a bundle.";  
        leaf notification-record {  
          type notification-record-ref;  
          description  
            "Included notification within the bundle. Full records  
            or elements of this record might not be sent to any  
            included receiver based on security permissions for that  
            receiver.";  
        }  
      }  
    }  
  }  
  container receiver-notification-messages {  
    description  
      "Contains instances of messages generated for a specific  
      receiver.";  
    list receiver-notification-message {  
      description  
        "Maintains instances of bundled messages targeted for a  
        receiver.";  
      leaf receiver {  
        type receiver-ref;  
        description  
          "Reference to the recipient targeted for this bundled  
          notification message. (As a receiver is unique to a
```

```

        subscription, this also identifies the subscription
        explicitly. If something other than receiver is used, a
        method to identify the subscription is also needed as it
        can't automatically be derived from the notification
        record.";
    }
    uses notification-message-receiver-header {
        refine notification-time {
            mandatory true;
        }
    }
    leaf record-count {
        type uint16;
        description
            "Number of records actually sent to a receiver after
            considering the application of NACM policies on the
            notification records.";
    }
    container included-notification-records {
        description
            "Contains the records sent to a receiver within a
            specific notification message.";
        list notification-record {
            description
                "Maintains instances of records sent to a receiver.";
            anydata receiver-record-contents {
                mandatory true;
                description
                    "The specific security filtered contents of one
                    record going to a receiver.";
            }
        }
    }
}
<CODE ENDS>
```

## 8. Security Considerations

to be populated

## 9. References

### 9.1. 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, <<http://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, <<http://www.rfc-editor.org/info/rfc7950>>.

### 9.2. Informative References

- [sn] Voit, E., Clemm, A., Gonzalez Prieto, A., Prasad Tripathy, A., and E. Nilsen-Nygaard, "Subscribing to Event Notifications", February 2017, <<https://datatracker.ietf.org/doc/draft-ietf-netconf-subscribed-notifications/>>.
- [yang-push] Clemm, A., Voit, E., Gonzalez Prieto, A., Prasad Tripathy, A., and E. Nilsen-Nygaard, "Subscribing to YANG datastore push updates", February 2017, <<https://datatracker.ietf.org/doc/draft-ietf-netconf-yang-push/>>.

## Appendix A. Issues being worked

(To be removed by RFC editor prior to publication)

We need to define the ways to invoke and configure the capability within an originating device. This includes defining what header types are selected.

Should we allow multiple subscriptions to be associated with an update record via a leaf-list?

Should the subscription id in a notification actually be a leafref?

We need to do a lot more to discuss transport efficiency implications.

Authors' Addresses

Eric Voit  
Cisco Systems  
  
Email: [evoit@cisco.com](mailto:evoit@cisco.com)

Andy Bierman  
YumaWorks  
  
Email: [andy@yumaworks.com](mailto:andy@yumaworks.com)

Alexander Clemm  
Huawei  
  
Email: [ludwig@clemm.org](mailto:ludwig@clemm.org)

Tim Jenkins  
Cisco Systems  
  
Email: [timjenki@cisco.com](mailto:timjenki@cisco.com)

NETCONF Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: September 14, 2017

Z. Wang  
G. Zheng  
Huawei Technologies  
March 13, 2017

Network Configuration Protocol (NETCONF) Proxy  
draft-wangzheng-netconf-proxy-00

Abstract

This document presents Network Configuration Protocol (NETCONF) Proxy through which NETCONF requests can be forwarded to a target host. It would be useful when a client does not have direct network access to a target host.

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 <http://datatracker.ietf.org/drafts/current/>.

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

This Internet-Draft will expire on September 14, 2017.

Copyright Notice

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

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

## Table of Contents

1. Introduction . . . . .	2
1.1. Motivation . . . . .	2
1.2. Requirements Terminology . . . . .	3
2. Solution Overview . . . . .	3
3. The NETCONF Client . . . . .	5
4. The Proxy . . . . .	6
5. The Target . . . . .	7
6. New attribute: target-id . . . . .	8
7. YANG DATA MODEL . . . . .	8
7.1. Overview . . . . .	8
7.2. YANG Module . . . . .	9
8. Security Considerations . . . . .	11
9. IANA Considerations . . . . .	11
10. Normative References . . . . .	11
Authors' Addresses . . . . .	11

## 1. Introduction

This document proposes a NETCONF Proxy mechanism. The mechanism extends the NETCONF protocol [RFC6241] which provides a standard way to set up NETCONF session. At its core, the mechanism defined here introduces a set of new operations which allow a client to forward NETCONF requests to a target host through an intermediary NETCONF proxy server, especially in case where client would otherwise not have direct network access to a target host. The document also includes YANG data model which extend the model and RPCs defined within [RFC6241].

## 1.1. Motivation

NETCONF provide a RPC-based mechanism to facilitate communication between a client and a server. The client can be a script or application typically running as part of a network manager. The server is typically a network device [RFC6241]. However, the network manager may not have direct network access to the target network devices. For example, some target network devices may locate in a network with private addresses behind a NAT device or a firewall. Thus, network manager cannot direct communicate with these target devices.

NETCONF Call Home [RFC8071] provides a mechanism that allows NETCONF Servers to initiate a connection with a NETCONF client, reversing the normal direction of NETCONF session setup. This allows a NETCONF Server, e.g. a networking device that needs to be managed, to reach out to a NETCONF Client, e.g. an Operations Support System of an SDN controller, in order to be managed. By reversing the direction in

which NETCONF sessions are normally set up, problems such as establishing connectivity with devices behind a firewall can be alleviated. However, NETCONF Call Home requires that the server knows its client by way of configuration or discovery. It does not address the scenarios as presented below:

1. In some NFV scenarios, VNF instances are running in a private network. To reduce the management resources (like IP resources, bandwidth, etc) of large-scale management activities, these VNF instances may not be assigned IP addresses. Then the element management system (EMS), which located in public network, cannot be aware of the addresses of VNF instances. Therefore, the element management system (EMS) is difficult to manage these VNF instances via NETCONF protocol.
2. And in some cloud network scenarios, the gateway network element (GNE) and non-gateway network elements (N-GNEs) communicate with each other using some private protocol. And these non-gateway network elements (N-GNEs) may not IP devices. Therefore, the cloud centre EMS (element management system) cannot be aware of the addresses of N-GNEs. Thus, the element management system (EMS) is difficult to manage these N-GNE devices via NETCONF protocol.

To solve that problem, this document proposes a NETCONF Proxy mechanism. The proxy can acts as an intermediary between manager and target device, therefore the client can set up a NETCONF session to a target through a NETCONF Proxy.

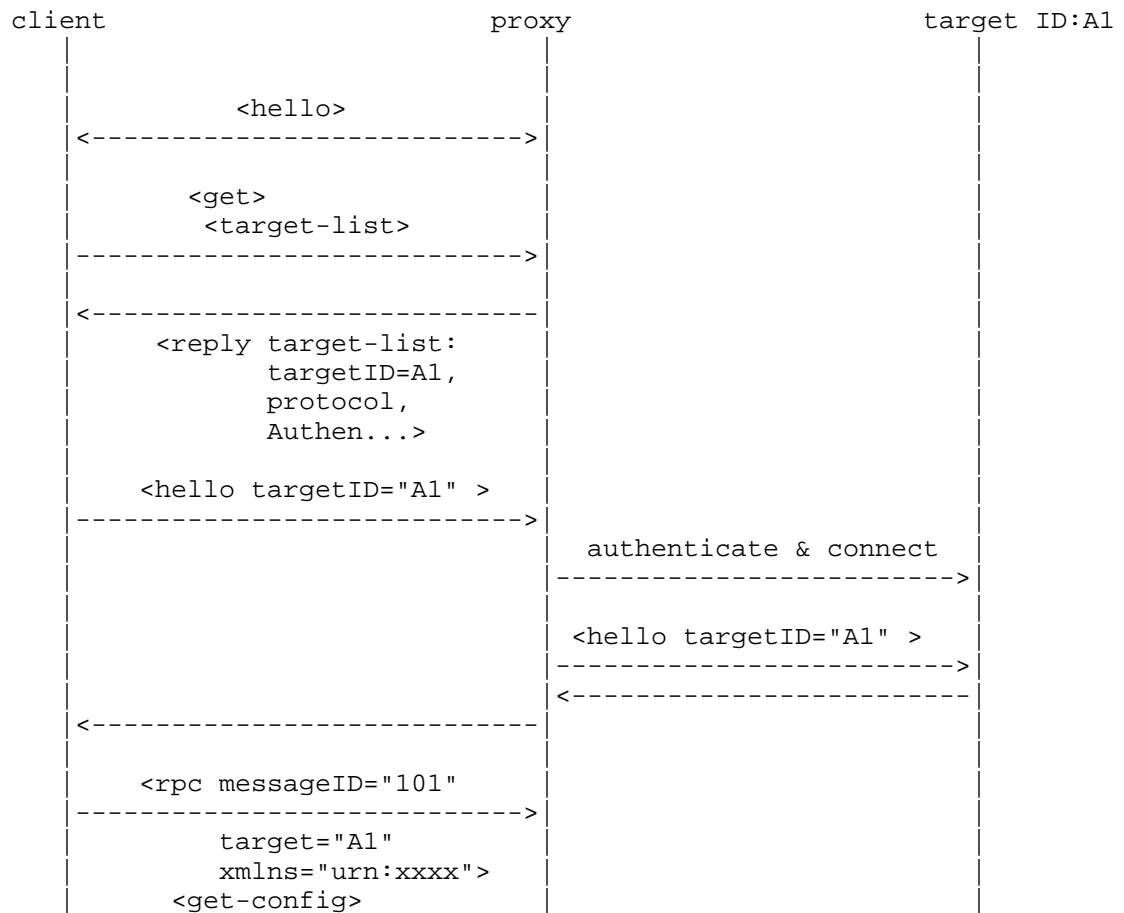
The mechanism allows the client to subsequently direct NETCONF requests to the server, to receive responses, and to subscribe to notifications from the server. While the NETCONF Proxy can be used to traverse NAT boundaries, it should be noted that it does not apply NAT mappings for contents carried as part of the NETCONF payload; specifically, it does not substitute IP address information that is carried as part of data nodes.

## 1.2. Requirements 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].

## 2. Solution Overview

The diagram below illustrates how the client can set up a NETCONF session to a target through the NETCONF proxy.



This diagram makes the following points:

1. The client initiates the connection using the SSH/TLS transport protocol. When the NETCONF session is established, the client and proxy MUST send a `<hello>` element containing a list of that peer's capabilities. The proxy SHOULD send at least the "netconf" and "proxy" capabilities. And other rules of capabilities exchange described in section 8 of [RFC6241].
2. The client sends a `<get>` RPC to proxy to retrieve the "target-list" of the proxy.
3. The proxy responds with a `<get-reply>` RPC which containing "target-list" attributes. The "target-list" attributes includes



the target's information such as target-id, protocol, authentication, etc.

4. The client receive a <get-reply> RPC from the proxy, and looks up the value of "target-list". Then the client constructs a <hello> message according to the received "target-list". This <hello> message SHOULD contain at least a "target-id" attribute. And then client sends this <hello> message to proxy and waits for a reply.

5. The proxy receives the <hello> message and checks the value of "target-id" attribute:

If the target is not found, then an "invalid-target" error will be returned.

If the target can be found, then the proxy initiates a connection to corresponding target. And then proxy forwards the <hello> message, which received from client, to corresponding target.

6. The target receives the <hello> message and then responds a <hello> message containing a list of capabilities. The rules of capabilities exchange described in section 8 of [RFC6241].
7. Now, client established a NETCONF session to a target through NETCONF Proxy. Subsequently, the client can direct NETCONF requests to the target, to receive responses, and to subscribe to notifications from the target.

### 3. The NETCONF Client

The term "client" is defined in [RFC6241], Section 1.1 "client". In the context of network management, the NETCONF client might be a network management system for example a EMS (element management system).

The client operation describes as follows:

1. The client initiates a connection to proxy using the SSH/TLS transport protocol [RFC6242]. How to establish an SSH/TLS transport connection is described in [RFC6242]
2. When the NETCONF session is established, the client sends a <hello> message to proxy, then waits for a reply. This <hello> message contains a list of client's capabilities.

3. After capabilities exchange, the client sends a <get> RPC to proxy to retrieve the "target-list" of the proxy, then waits for a reply.

4. The client receive a <get-reply> RPC from the proxy, looks up the value of "target-list", and then constructs a <hello> message according to the received "target-list". This <hello> message SHOULD contain at least a "target-id" attribute. For example, if the client received "target-list" containing "target-id=A1", then the client constructs <hello> message:

```
C: <hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">  
C:   <target-id>A1</target-id>  
C:   <capability>foo</capability>  
C: </hello>
```

And then client sends this <hello> message to proxy and waits for a reply.

5. If the reply presents the "capabilities" of target, the proxy connection is established. Subsequently, the client can direct NETCONF requests to the target, to receive responses, and to subscribe to notifications from the target.

6. If the reply contains the "invalid-target" error, the process turn to step (4) or aborts.

7. Otherwise, the client interprets the error and aborts.

#### 4. The Proxy

The Proxy should ensure that requests given by client for a particular target device should reach the target device and the operations should be executed on that target device and the response should be given back to the client.

The proxy operation describes as follows:

1. When the NETCONF session is established, the proxy sends a <hello> element containing a list of proxy's capabilities. The proxy SHOULD send at least the "netconf" and "proxy" capabilities. And other rules of capabilities exchange described in section 8 of [RFC6242].

2. The proxy receives the <get> RPC and then responds with a <get-reply> RPC which containing "target-list" attributes. The "target-list" attributes SHOULD includes the target's information such as target-id, protocol, authentication, etc. The following example shows a "target-list":

```
<target-list>
  <target-id>A1</target-id>
  <protocol>protocol-foo</protocol>
  <authentication>authentication-foo</authentication>
</target-list>
```

3. The proxy receives the <hello> message and checks the value of "target-id" attribute:

If the target is not found in target-list, then an "invalid-target" error will be returned.

If the target can be found in target-list, the proxy checks the corresponding "protocol" and "authentication" of the "target-id". And then, the proxy uses corresponding protocol to establish a connection to the target. After session established, the proxy forwards the <hello> message, which received from client, to corresponding target.

4. If the reply presents the "capabilities" of target, the proxy connection is established. Subsequently, the proxy transports the messages received from the client to the target and vice versa.

## 5. The Target

The term "target" is equivalent to the term "server" which is defined in [RFC6242], Section 1.1 "server". In the context of network management, the target is typically a network device.

The target operations describes as follows:

1. If the connection between the proxy and the target established. And target receives the <hello> message from the proxy, and then responds a <hello> message containing a list of capabilities. The rules of capabilities exchange described in section 8 of [RFC6242].

2. After client established a NETCONF session to a target through NETCONF Proxy. The client sends a series of one or more RPC request messages, which cause the server to respond with a corresponding series of RPC reply messages.

## 6. New attribute: target-id

A proxy can be used by a client to connect to several servers and to maintain multiple NETCONF sessions. A client may use the proxy even to maintain multiple NETCONF sessions with the same NETCONF server. When issuing a NETCONF request, a client must therefore differentiate between NETCONF sessions. To solve this problem, a new attribute "target-id" is defined. This attribute allow the proxy to forward RPC to corresponding target.

For example:

The following <rpc> element invokes the NETCONF <get> method and includes the "target-id" attribute:

```
<rpc message-id="101"
      target-id="A1"
      xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get/>
</rpc>

<rpc-reply message-id="101"
            target-id="A1"
            xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <!-- contents here... -->
  </data>
</rpc-reply>
```

## 7. YANG DATA MODEL

### 7.1. Overview

The YANG data model for NETCONF Proxy is depicted in the following figure. Following Yang tree convention in the depiction, brackets enclose list keys, "rw" means configuration, "ro" operational state data, "?" designates optional nodes, "\*" designates nodes that can have multiple instances. A "+" at the end of a line indicates that the line is to be concatenated with the subsequent line.

```
module: ietf-netconf-proxy
  +--rw proxy {proxy}?
    +--rw proxy-name?      string
    +--rw target-list* [target-id]
      +--rw target-id      string
      +--rw protocol?     string
      +--rw authentication? string
```

## 7.2. YANG Module

```
<CODE BEGINS> file "ietf-netconf-proxy@2017-03-09.yang"
module ietf-netconf-proxy {

  namespace "urn:ietf:params:xml:ns:yang:ietf-netconf-proxy";

  prefix np;

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

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

    WG Chair: Mehmet Ersue
              mehmet.ersue@nsn.com

    Editor:   zitao wang
              wangzitao@huawei.com";

  description
    "NETCONF Protocol Data Types and Protocol Operations.

    Copyright (c) 2011 IETF Trust and the persons identified as
    the document authors.  All rights reserved.

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

    This YANG module describe how to define a netconf proxy";

  revision 2017-03-09 {
    description
```

```
        "Initial revision";
    reference
        "draft-wang-netconf-proxy";
}

feature proxy {
    description
        "Netconf proxy";
}

container proxy {
    if-feature proxy;
    leaf proxy-name{
        type string;
        description
            "Proxy name";
    }
    list target-list {
        key "target-id";
        leaf target-id{
            type string;
            description
                "Target ID";
        }
        leaf protocol {
            type string;
            description
                "Support protocols";
        }
        leaf authentication {
            type string;
            description
                "Authentication";
        }
        description
            "List for target information";
    }
    description
        "Container for NETCONF Proxy";
}

}
<CODE ENDS>
```

## 8. Security Considerations

The security considerations described in [RFC6242] and [RFC7589], and by extension [RFC4253], [RFC5246] apply here as well.

## 9. IANA Considerations

TBD

## 10. 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, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC4253] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, DOI 10.17487/RFC4253, January 2006, <<http://www.rfc-editor.org/info/rfc4253>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<http://www.rfc-editor.org/info/rfc5246>>.
- [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, <<http://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, <<http://www.rfc-editor.org/info/rfc6242>>.
- [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, <<http://www.rfc-editor.org/info/rfc7589>>.
- [RFC793] Postel, J., "TRANSMISSION CONTROL PROTOCOL", STD 7, September 1981, <<https://www.ietf.org/rfc/rfc793.txt>>.

## Authors' Addresses

Zitao Wang  
Huawei Technologies  
101 Software Avenue, Yuhua District  
Nanjing  
China

EMail: wangzitao@huawei.com

Guangying Zheng  
Huawei Technologies  
101 Software Avenue, Yuhua District  
Nanjing  
China

EMail: zhengguangying@huawei.com