SACM Working Group H. Birkholz
Internet-Draft Fraunhofer SIT
Intended status: Informational T. Zhou
Expires: April 21, 2018 Huawei
X. Liu
Jabil
E. Voit
Cisco Systems
October 18, 2017

YANG Push Operations for CoMI
draft-birkholz-yang-push-coap-problemstatement-00

Abstract

This document provides a problem statement, derives an initial gap analysis and illustrates a first set of solution approaches in regard to augmenting YANG data stores based on the CoAP Management Interface with YANG Push capabilities. A binary transfer mechanism for YANG Subscribed Notifications addresses both the requirements of constrained-node networks and the need for semantic interoperability via self-descriptiveness of the corresponding data in motion.

Table of Contents

1. Context of the Problem

A binary transfer capability for YANG Subscribed Notifications [I-D.ietf-netconf-subscribed-notifications] based on YANG Push [I-D.ietf-netconf-yang-push] can be realized by using existing RFC and I-D work as building blocks. This section is intended to provide a corresponding overview of the existing ecosystem in order to identify gaps and therefore provide a problem statement.

1.1. Binary YANG transfer protocol

The CoAP Management Interface I-D (CoMI [I-D.ietf-core-comi]) defines operations for a YANG data store based on the Constrained Application Protocol (CoAP [RFC7252]). CoAP uses a request/response interaction model that is based on HTTP (similar to RESTCONF [RFC8040]) and allows for multiple transports, including UDP or TCP (see [I-D.ietf-core-coap-tcp-tls]). The Concise Binary Object Representation (CBOR [RFC7049]) is used for the serialization of data in motion in respect to CoAP operations and the data modeled with YANG [I-D.ietf-core-yang-cbor].

1.2. Device-Type Scope

[I-D.ietf-core-comi] states that CoAP "is designed for Machine to Machine (M2M) applications such as smart energy, smart city and building control. Constrained devices need to be managed in an automatic fashion to handle the large quantities of devices that are expected in future installations. Messages between devices need to be as small and infrequent as possible. The implementation complexity and runtime resources need to be as small as possible."

In addition, [I-D.ietf-core-comi] highlights that "CoMI and RESTCONF are intended to work in a stateless client-server fashion. They use a single round-trip to complete a single editing transaction, where NETCONF needs up to 10 round trips. To promote small messages, CoMI uses a YANG to CBOR mapping [I-D.ietf-core-yang-cbor] and numeric identifiers [I-D.ietf-core-sid] to minimize CBOR payloads and URI length."

In essence, via CoMI, a small sensor can emit a set of measurements as binary encoded YANG notifications, which would only add a minimal overhead to the data in motion, but would increase interoperability significantly due to the powerful and widely used semantics enabled by YANG (in contrast to a set of raw values that always require additional context information and imperative guidance to be managed and post-processed appropriately).

1.3. Subscriptions via CoAP

The CoAP pub/sub I-D defines a CoAP Subscribe operation [I-D.ietf-core-coap-pubsub] that is based on observing resources via the Observe option for the GET operation as defined in [RFC7641]. The CoAP pub/sub draft is intended to provide the capabilities and characteristics of MQTT via a CoAP based protocol. The only other CoAP operation that supports the Observe option is the FETCH operation defined in [RFC8132].

The Observe option creates a small corresponding state on the server side that eliminates the need for continuous polling of a resource via subsequent requests. Instead, subsequent responses including both the Observe option and using the token of the request that initiated the observation are returned when the observed resource changes. A subscription (i.e. the observe state retained on the server) can be discarded by the client by sending a correspond CoAP GET with Observe using an Observe parameter of 1 or simply by "forgeting" the observation and return a CoAP Reset after receiving a notification in the context of the subscription. A subscription can also be discarded by the server by sending a corresponding response that does not contain an Observe option.

The subscription used in CoAP pub/sub are used to subscribe to a topic provided by a CoAP broker REST API. YANG Push [I-D.ietf-netconf-yang-push] and corresponding YANG Subscribed Notifications are used to subscribe to data node updates provided by a YANG management interface. YANG subscriptions can include a filter expression (either a subtree expression or an XPATH expression). The encoding rules of XPATH expressions in CBOR are covered by [I-D.ietf-core-yang-cbor].

1.4. Configured Subscriptions and Call-Home

Configured subscriptions are basically static configuration that creates subscription state on the YANG data store when it is started and persists between boot-cycles without the need of a client to create that subscription state. In consequence, a configured subscription can result in unsolicited pushed notifications in respect to a YANG client.

A popular variant of the configured subscription as defined in [I-D.ietf-netconf-yang-push] is the Call Home procedure defined in [RFC8071]. In this approach, a Transport Layer application association with the YANG client is initiated by the YANG data store. After this "initial phase, in which the YANG server is acting like a client", the existing Transport Layer connection (or session, in case of, for example, TLS) is then used to the YANG client to initiate a subscription (i.e. the YANG client is initiating a dynamic subscription based on a pre-configured request retained and issued by the YANG data store).

1.5. Bootstrapping of Drop-Shipped Pledges

[I-D.ietf-anima-bootstrapping-keyinfra] highlights that effectively "to literally ’pull yourself up by the bootstraps’ is an impossible action. Similarly, the secure establishment of a key infrastructure without external help is also an impossibility."

According to [I-D.ietf-anima-bootstrapping-keyinfra] the bootstrapping approach Call-Home has problems and limitations, which (amongst others) the draft itself is trying to address:

- o the pledge requires realtime connectivity to the vendor service
- o the domain identity is exposed to the vendor service (this is a privacy concern)
- o the vendor is responsible for making the authorization decisions (this is a liability concern)

A Pledge in the context of [I-D.ietf-anima-bootstrapping-keyinfra] is "the prospective device, which has an identity installed by a third-party (e.g., vendor, manufacturer or integrator)."

A Pledge can be "drop-shipped", which refers to "the physical distribution of equipment containing the 'factory default' configuration to a final destination. In zero-touch scenarios there is no staging or pre-configuration during drop-ship."

In the scope of Call-Home as a part of YANG Push, either the factory default configuration of a drop-shipped Pledge that is a YANG data store would require to include the "home to Call Home" configuration or it has to be configured locally.

[I-D.ietf-netconf-zerotouch] is intended to provide more flexibility to the Call-Home procedure already - by allowing to stage connection attempts to a locally administered network and if that fails fall back to connecting to a remotely administered network. Alas, [I-D.ietf-netconf-zerotouch] is either prone to the same limitations as cited above or requires local configuration in order to find the home to Call-Home.

The "Join Registrar" defined by [I-D.ietf-anima-bootstrapping-keyinfra] mitigates the cited problems and limitation by introducing "a representative of the domain that is configured, perhaps autonomically, to decide whether a new device is allowed to join the domain. The administrator of the domain interfaces with a Join Registrar (and Coordinator) to control this process. Typically a Join Registrar is "inside" its domain."

2. Summary of the Problem Statement

Currently, the following gaps are identified:

- o no CoAP Subscribe procedure for dynamic YANG subscriptions is standardized that is able to convey a filter expression and potentially other metadata required in the context of a YANG Subscribed Notifications application association. Analogously, new payload types (e.g. a FETCH payload media-type) have to be defined.

- no CoAP Call Home feature is standardized to support a popular variant of configured YANG subscriptions.

- no general Call Home mechanism is standardized that enables the discovery of "a home to Call Home" or that would be able to deal with "changing homes" in a dynamic but secure manner.

In addition to the identified gaps, the semantics of metadata - if there are any - that have to be conveyed to or from a YANG data store in order to subscribe to a (filtered) YANG module or data node are not identified.

The problem statement could be summarized as follows:

"There is no complete solution based on CoAP to enable a freshly unpacked YANG data store ("drop-shipped pledge", e.g. the cliche light bulb) to discover an appropriate home it can than Call-Home to in a secure and trusted manner in order to push (un-)solicited subscribed notifications."

3. Potential Approaches and Solutions

There are multiple approaches that could lead to viable solutions that address the identified gaps. The following sections illustrate the general solution context and some of the most promising approaches.

3.1. YANG subscription variants

A YANG Push update subscription service both provides support for dynamic subscription (i.e. subscription state created by a client request, allowing for solicited push notifications in the context of an up-time cycle of the server) and configured subscription (i.e. subscription configuration retained on the server, allowing for unsolicited push notifications across up-time cycles of the server).

3.2. YANG Push via CoAP

The two CoAP operations that enable a subscription mechanism are GET and FETCH (i.e. by supporting the Observe option). Both operations are viable candidates for creating a CoAP-based YANG Push mechanism for CoMI.

3.3. Dynamic Subscriptions

Using CoAP, the client issuing the initial subscription request creates the subscription state. Examples are the GET or FETCH operation including an Observe option using an Observe parameter of 0 (zero).

3.3.1. YANG Push via GET

This usage scenario requires two consecutive operations. It is not possible to transfer a filter expression included in a GET operation. In consequence, a POST operation on a collection resource has to be conducted in order to convey a filter expression to the YANG data store, allowing it to return an URI that contains the data node information filtered in respect to the posted filter expression (encoded in CBOR).

This variant allows for multiple clients to observe a specific filtered data node without conducting a POST operation, if the corresponding URI is made known to other clients that did not conduct the POST operation or, for example, is canonically linked to/ derivable from a filter expression.

3.3.2. YANG Push via FETCH

This usage scenario requires only one operation. A FETCH operation can include a body that is capable to contain a filter expression and potentially other metadata that might be required to establish a suitable subscription state on the YANG data store.

It might be possible that this variant could introduce a slight delay in respect to response time if providing a filtered resource requires a lot of computation time on a constrained device. I.e. the resource cannot be prepared "beforehand".

3.4. Configured Subscriptions

Using CoAP, the server retains configuration that creates subscription state when the YANG data store is started. The client has to have or gain knowledge of the CoAP tokens that are included in the responses created in the context of the subscription state create from server configuration.

3.4.1. Retaining the Content of a GET Operation as Configuration

This usage scenario "mimics" the receiving of a subscription request by storing the corresponding information that are relevant for creating a subscription state as configuration on the YANG data store. I.e. the configuration would be including the YANG client IP address and the CoAP token to be used in the responses that convey the subscribed notifications.

This variant requires that the client also knows or gains knowledge of the corresponding CoAP token in order to not discard the incoming responses.

3.4.2. Call Home via CoAP

This usage scenario defines the Call Home procedure standardized in [RFC8071] as an additional capability of CoAP. DTLS or TLS state is initiated by the YANG data store and triggers a dynamic subscription procedure of the YANG client using the session initiated by the YANG data store.

3.4.3. Dynamic Home Discovery

This usage scenario is based on the Bootstrapping Remote Secure Key Infrastructures I-D [I-D.ietf-anima-bootstrapping-keyinfra] and EST over secure CoAP I-D [I-D.vanderstok-ace-coap-est] and requires the standardization of a general use of Join Registrars in the context of YANG data stores that support YANG Push via static subscriptions.

4. IANA considerations

This document includes no requests to IANA, but solutions drafts incubated via this document might.

5. Security Considerations

This document includes no security considerations, but solution drafts incubated via this document will.

6. Acknowledgements

Carsten Bormann, Klaus Hartke, Michel Veillette

7. Change Log

First version -00

8. Normative 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-08 (work in progress), October 2017.

[I-D.ietf-core-coap-pubsub]
Koster, M., Keranen, A., and J. Jimenez, "Publish-Subscribe Broker for the Constrained Application Protocol (CoAP)", draft-ietf-core-coap-pubsub-02 (work in progress), July 2017.

[I-D.ietf-core-coap-tcp-tls]
Bormann, C., Lemay, S., Tschofenig, H., Hartke, K., Silverajan, B., and B. Raymor, "CoAP (Constrained Application Protocol) over TCP, TLS, and WebSockets", draft-ietf-core-coap-tcp-tls-09 (work in progress), May 2017.

[I-D.ietf-core-comi]
Veillette, M., Stok, P., Pelov, A., and A. Bierman, "CoAP Management Interface", draft-ietf-core-comi-01 (work in progress), July 2017.

[I-D.ietf-core-sid]
Veillette, M., Pelov, A., Turner, R., Minaburo, A., and A. Somaraju, "YANG Schema Item iDentifier (SID)", draft-ietf-core-sid-01 (work in progress), May 2017.

[I-D.ietf-core-yang-cbor]
Veillette, M., Pelov, A., Somaraju, A., Turner, R., and A. Minaburo, "CBOR Encoding of Data Modeled with YANG", draft-ietf-core-yang-cbor-05 (work in progress), August 2017.

[I-D.ietf-netconf-subscribed-notifications]
Voit, E., Clemm, A., Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Custom Subscription to Event Notifications", draft-ietf-netconf-subscribed-notifications-05 (work in progress), October 2017.

[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-10 (work in progress), October 2017.

[I-D.ietf-netconf-zerotouch]
Watsen, K., Abrahamsson, M., and I. Farrer, "Zero Touch Provisioning for NETCONF or RESTCONF based Management", draft-ietf-netconf-zerotouch-17 (work in progress), September 2017.

[I-D.vanderstok-ace-coap-est]
Kumar, S., Stok, P., Kampanakis, P., Furuhed, M., and S. Raza, "EST over secure CoAP (EST-coaps)", draft-vanderstok-ace-coap-est-02 (work in progress), June 2017.

[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>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/info/rfc7252>.

[RFC7641] Hartke, K., "Observing Resources in the Constrained Application Protocol (CoAP)", RFC 7641, DOI 10.17487/RFC7641, September 2015, <https://www.rfc-editor.org/info/rfc7641>.

[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>.

[RFC8132] van der Stok, P., Bormann, C., and A. Sehgal, "PATCH and FETCH Methods for the Constrained Application Protocol (CoAP)", RFC 8132, DOI 10.17487/RFC8132, April 2017, <https://www.rfc-editor.org/info/rfc8132>.

Authors' Addresses

Henk Birkholz
Fraunhofer SIT
Rheinstrasse 75
Darmstadt 64295
Germany

Email: henk.birkholz@sit.fraunhofer.de

Tianran Zhou
Huawei
156 Beiqing Rd.
Beijing, Haidian District
China

Email: zhoutianran@huawei.com

Xufeng Liu
Jabil
8281 Greensboro Drive, Suite 200
McLean VA 22102
USA

Email: Xufeng_Liu@jabil.com

Eric Voit
Cisco Systems

Email: evoit@cisco.com

Network Working Group I. Bryskin
Internet-Draft Huawei Technologies
Intended status: Informational X. Liu
Expires: April 19, 2018 Jabil
A. Clemm
Huawei
H. Birkholz
Fraunhofer SIT
T. Zhou
Huawei
October 16, 2017

YANG PUSH Based Generalized Network Control Automation Problem Statement
draft-bryskin-netconf-automation-framework-00

Abstract

This document describes the objective of the YANG PUSH based generalized network control automation framework.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

YANG "Custom Subscription to Event Notifications" model [I-D.ietf-netconf-subscribed-notifications] allows for a network client automation of network remote monitoring. Specifically, using this model, a network client can subscribe on and receive one or more data streams, each associated with one or more events defined by YANG model(s) governing the network's YANG data store(s). The client can also tailor said streams to its needs by specifying filters on the streams contents, but, otherwise, the client has no control on the stream contents. For example, the client has no way of expanding a stream to carry additional information that was not defined to be a part of said stream.

YANG "Subscribing to YANG datastore push updates" model [I-D.ietf-netconf-yang-push], which is an augmentation of the "Custom Subscription to Event Notifications" model, defines a higher level of network remote monitoring automation - it allows for the client itself to define the origins, trigger/maintain conditions and contents of data streams to be sent by the network to the client. This capability is modeled via target-trigger-notify constructs, which allow for the client to specify data store nodes of interest and, possibly, sub-trees rooted by them (targets), conditions to trigger and maintain associated with them streams (e.g. particular change(s) in one or more of the nodes attributes), the contents of the streams and filters to further fine-tune the streams according to the client's needs.

It could be observed that the notify part of the target-trigger-notify construct stands for "send me notification', which is one of, generally speaking, many actions the client might want the network to perform, provided that the target-trigger condition holds. For example, instead of sending a notification with some pre-denied content, the client might want the network to perform:

a. immediate network re-configuration (e.g. modification of one or more attributes of one or more CONFIG=TRUE data store nodes);

b. scheduling one time or periodic such reconfigurations in the future;

c. calling an RPC defined by one of the YANG models supported by the network (e.g. calling network's path computer to evaluate whether an alternative/more optimal path is available for a given connection);

d. Dynamic linking/unlinking parent and child data stores supported by the network;

e. etc.

It could also be observed that "periodic" and "on-change" are two of the conditions that the client might want. The conditions can be expanded to be a logical expression of other event states and some operational data states of the network., as well, as outputs of RPCs.

2. Objective

The main objective of the YANG PUSH Based Generalized Network Control Automation framework is to generalize the target-trigger-notify construct into event-condition-action construct, where:

event
: a particular change in the network state explicitly defined by one of the YANG models supported by the network or implicitly defined by the client, which is constantly monitored by the network;

condition
: a logical expression that is evaluated only once after the associated event is detected;

action
: an operation (non-exhaustive list of which is described above) to be carried out by the network when the associated event is detected and the associated condition is met.

The client will be able to describe the desired network behavior by configuring with the network event-condition-action triplets as rules prior to any services provided by the network to the client. Such an approach will take the client out of the network control loop, thus, changing the client’s role from being network’s "micro-manager" to being network’s "police officer", which interferes into network operations only in exceptional/unpredicted situations.

There are numerous benefits to such paradigm, including:

- o lower latency, faster responsiveness of the network to various events/conditions;

- o better scale (e.g. the client may control more networks because it does not have to monitor/micro-manage any of them);

- o CPU and bandwidth savings due to the reduced amount of communication between the client and the network.

It is envisioned that the YANG PUSH Based Generalized Network Control Automation framework will fit well within "SUPA Policy-based Management Framework" [I-D.ietf-supa-policy-based-management-framework], which will inherently provide a higher level of automation, for example, by:

a. combining multiple micro-conditions into a single macro-condition via a number of logical operations;

b. combining multiple micro-actions into a single transaction with a possibility of specifying policies with respect to handling errors/exceptions of each of the transaction components.

3. IANA Considerations

This document has no actions for IANA.

4. Security Considerations

This document does not define networking protocols and data, hence are not directly responsible for security risks.

5. Acknowledgements

6. References

6.1. Normative References

[I-D.ietf-netconf-subscribed-notifications]
Voit, E., Clemm, A., Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Custom Subscription to Event Notifications", draft-ietf-netconf-subscribed-notifications-05 (work in progress), October 2017.

[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-10 (work in progress), October 2017.

[I-D.ietf-supa-policy-based-management-framework]
LIU, W., Xie, C., Strassner, J., Karagiannis, G., Klyus, M., and J. Bi, "SUPA Policy-based Management Framework", draft-ietf-supa-policy-based-management-framework-03 (work in progress), July 2017.

6.2. Informative References

[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>.

[I-D.ietf-supa-generic-policy-data-model]
Halpern, J. and J. Strassner, "Generic Policy Data Model for Simplified Use of Policy Abstractions (SUPA)", draft-ietf-supa-generic-policy-data-model-04 (work in progress), June 2017.

[I-D.ietf-supa-generic-policy-info-model]
Strassner, J., Halpern, J., and S. Meer, "Generic Policy Information Model for Simplified Use of Policy Abstractions (SUPA)", draft-ietf-supa-generic-policy-info-model-03 (work in progress), May 2017.

Authors' Addresses

Igor Bryskin
Huawei Technologies

EMail: Igor.Bryskin@huawei.com

Xufeng Liu
Jabil

EMail: Xufeng_Liu@jabil.com

Alexander Clemm
Huawei

EMail: ludwig@clemm.org

Henk Birkholz
Fraunhofer SIT

EMail: henk.birkholz@sit.fraunhofer.de

Tianran Zhou
Huawei

EMail: zhoutianran@huawei.com

Network Working Group   A. Clemm
Internet-Draft   Y. Qu
Intended status: Informational   J. Tantsura
Expires: May 3, 2018   Futurewei Technologies, Inc.
                                                        October 30, 2017

Discrepancy detection between NMDA datastores
draft-clemm-netconf-nmda-diff-01

Abstract

This document defines a capability that allows to report discrepancies between management datastores in Netconf or Restconf servers that comply with the NMDA architecture. The capability is based on a set of RPCs that are defined as part of a YANG data model and that are intended to be used in conjunction with Netconf and Restconf.

Table of Contents

1. Introduction

The revised Network Management Datastore Architecture (NMDA) [NMDA] introduces a set of new datastores that each hold YANG-defined data [RFC7950] and represent a different "viewpoint" on the data that is maintained by a server. New YANG datastores that are introduced include <intended>, which contains validated configuration data that a client application intends to be in effect, and <operational>, which contains at least conceptually operational state data (such as statistics) as well as configuration data that is actually in effect.

NMDA introduces in effect a concept of "lifecycle" for management data, allowing to clearly distinguish between data that is part of a configuration that was supplied by a user, configuration data that has actually been successfully applied and that is part of the operational state, and overall operational state that includes both applied configuration data as well as status and statistics.

As a result, data from the same management model can be reflected in multiple datastores. Clients need to specify the target datastore to be specific about which viewpoint of the data they want to access. This way, an application can differentiate whether they are (for example) interested in the configuration that has been applied and is actually in effect, or in the configuration that was supplied by a client and that is supposed to be in effect.

Due to the fact that data can propagate from one datastore to another, it is possibly for discrepancies to occur. Some of this is entirely expected, as there may be a time lag between when a configuration is given to the device and reflected in <intended>, until when it actually takes effect and is reflected in <operational>. However, there may be cases when a configuration item that was to be applied may not actually take effect at all or needs an unusually long time to do so. This can be the case due to certain conditions not being met, resource dependencies not being resolved, or even implementation errors in corner conditions.

When configuration that is in effect is different from configuration that was applied, many issues can result. It becomes more difficult to operate the network properly due to limited visibility of actual status which makes it more difficult to analyze and understand what is going on in the network. Services may be negatively affected (for example, breaking a service instance resulting in service is not properly delivered to a customer) and network resources be misallocated.

Applications can potentially analyze any discrepancies between two datastores by retrieving the contents from both datastores and comparing them. However, in many cases this will be at the same time costly and extremely wasteful. It will also not be an effective approach to discover changes that are only "fleeting", or for that matter to distinguish between changes that are only fleeting from ones that are not and that may represent a real operational issue and inconsistency within the device.

This document introduces a YANG data model which defines RPCs, intended to be used in conjunction with NETCONF [RFC6241] or RESTCONF [RFC8040], that allow a client to request a server to compare two NMDA datastores and report any discepancies. It also features a dampening option that allows to exclude discrepancies that are only fleeting from the report.

2. Key Words

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

3. Definitions and Acronyms

NMDA: Network Management Datastore Architecture

RPC: Remote Procedure Call

4. Data Model Overview

At the core of the solution is a new management operation, <compare>, that allows to compare two datastores for the same data. The operation checks whether there are any discrepancies in values or in objects that are contained in either datastore, and returns any discrepancies as output. The output is returned in the format specified in YANG-Patch [RFC8072].

The YANG data model defines the <compare> operation as a new RPC. The operation takes the following input parameters:

- source: The source identifies the datastore that will serve as reference for the comparison, for example <intended>.

- target: The target identifies the datastore to compare against the source.

- filter-spec: This is a choice between different filter constructs to identify the portions of the datastore to be retrieved. It acts as a node selector that specifies which data nodes are within the scope of the comparison and which nodes are outside the scope. This allows a comparison operation to be applied only to a specific portion of the datastore that is of interest, such as a particular subtree. (The filter dow not contain expressions that would match values data nodes, as this is not required by most use cases and would complicate the scheme, from implementation to dealing with race conditions.)

- dampening: Identifies the minimum time period for which a discrepancy must persist for it to be reported. The reporting of the output MAY correspondingly delayed by the dampening period. Implementations MAY thus run a comparison when the RPC is first invoked, then wait until after the dampening period to check whether any differences still persist. This parameter is conditional of a dampening being supported as a feature.

The operation provides the following output parameter:

- differences: This parameter contains the list of differences, encoded per RFC8072, i.e. specifying which patches would need to be applied to the source to produce the target.

As part of the differences, it will be useful to include "origin" metadata where applicable, specifically when the target datastore is <operational>. This can help explain the cause of a discrepancy, for example when a data item is part of <intended> but the origin in <operational> is reported as "system". How to best report "origin" metadata is an item for further study, specifically whether it should be automatically returned per default or whether its reporting should be controlled using another RPC parameter.

The data model is defined in the ietf-nmda-compare YANG module. Its structure is shown in the following figure. The notation syntax follows [I-D.draft-ietf-netmod-yang-tree-diagrams].

```
module: ietf-nmda-compare

  rpcs:
    +---x compare
       +---w input
       |  +---w source            identityref
       |  +---w target            identityref
       |  +---w (filter-spec)?
       |  |  +--:(subtree-filter)
       |  |  |  +---w subtree-filter?   <anydata>
       |  |  +--:(xpath-filter)
       |  |     +---w xpath-filter?     yang:xpath1.0 {nc:xpath}?
       |  +---w dampening?        yang:timeticks {cmp-dampening}?
       +--ro output
          +--ro differences
```

Structure of ietf-nmda-compare

5. YANG Data Model

```
<CODE BEGINS> file "ietf-nmda-compare@2017-10-30.yang"
module ietf-nmda-compare {

  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-nmda-compare";

  prefix cp;

  import ietf-yang-types {
    prefix yang;
  }
  import ietf-datastores {
    prefix ds;
  }
  import ietf-yang-patch {
    prefix ypatch;
  }
  import ietf-netconf {
    prefix nc;
```

```
  }

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

     Author: Alexander Clemm
             <mailto:ludwig@clemm.org>

     Author: Yingzhen Qu
             <mailto:yingzhen.qu@huawei.com>

     Author: Jeff Tantsura
             <mailto:jefftant.ietf@gmail.com>";

  description
    "The YANG data model defines a new operation, <compare>, that
     can be used to compare NMDA datastores.";

  revision 2017-10-30 {
    description
      "Initial revision";
    reference
      "RFC XXXX: Discrepancy detection between NMDA datastores";
  }

  feature cmp-dampening {
    description
      "This feature indicates that the ability to only report
       differences that pertain for a certain amount of time,
       as indicated through a dampening period, is supported.";
  }

  /* RPC */
  rpc compare {
    description
      "NMDA compare operation.";
    input {
      leaf source {
        type identityref {
          base ds:datastore;
        }
        mandatory true;
        description
          "The source datastore to be compared.";
      }
      leaf target {
```

```
        type identityref {
          base ds:datastore;
        }
        mandatory true;
        description
          "The target datastore to be compared.";
      }
          choice filter-spec {
        description
          "Identifies the portions of the datastores to be
               compared.";

        anydata subtree-filter {
          description
            "This parameter identifies the portions of the
             target datastore to retrieve.";
          reference "RFC 6241, Section 6.";
        }
        leaf xpath-filter {
          if-feature nc:xpath;
          type yang:xpath1.0;
          description
            "This parameter contains an XPath expression
             identifying the portions of the target
             datastore to retrieve.";
        }
      }
      leaf dampening {
        if-feature cmp-dampening;
        type yang:timeticks;
        default "0";
        description
          "The dampening period, in hundredths of a second, for the
                   reporting of differences. Only differences that pertain
                   for at least the dampening time are reported.  Reporting
           of differences may be deferred by the dampening time.
           A value of 0 or omission of the leaf indicates no
           dampening.";
      }
    }
    output {
      container differences {
        uses ypatch:yang-patch;
        description
          "The list of differences, encoded per RFC8072.";
      }
    }
  }
```

```
}
<CODE ENDS>
```

6. IANA Considerations

6.1. Updates to 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-nmda-compare

Registrant Contact: The IESG.

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

6.2. Updates to the YANG Module Names Registry

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

name: ietf-nmda-compare

namespace: urn:ietf:params:xml:ns:yang:ietf-nmda-compare

prefix: cp

reference: RFC XXXX

7. Security Considerations

Comparing discrepancies between datastores requires a certain amount of processing resources at the server. An attacker could attempt to attack a server by making a high volume of discrepancy detection requests. Server implementations can guard against such scenarios in several ways. For one, they can implement NACM in order to require proper authorization for requests to be made. Second, server implementations can limit the number of requests that they serve in any one time interval, potentially rejecting requests made at a higher frequency than the implementation can reasonably sustain.

8. Acknowledgments

We thank Rob Wilton for valuable feedback and suggestions on an earlier revision of this document.

9. Normative References

[I-D.draft-ietf-netmod-yang-tree-diagrams]
Bjorklund, M. and L. Berger, "YANG Tree Diagrams", I-D draft-ietf-netmod-yang-tree-diagrams, June 2017.

[NMDA] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture", October 2017, <https://datatracker.ietf.org/doc/draft-ietf-netmod-revised-datastores/>.

[notif-sub]
Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Custom subscription to event notifications", October 2017, <https://datatracker.ietf.org/doc/draft-ietf-netconf-subscribed-notifications/>.

[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>.

[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>.

[yang-push] Clemm, A., Voit, E., Gonzalez Prieto, A., Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "Subscribing to YANG datastore push updates", October 2017, <https://datatracker.ietf.org/doc/draft-ietf-netconf-yang-push/>.

Authors' Addresses

Alexander Clemm
Futurewei Technologies, Inc.
2330 Central Expressway
Santa Clara, CA 95050
USA

Email: ludwig@clemm.org

Yingzhen Qu
Futurewei Technologies, Inc.
2330 Central Expressway
Santa Clara, CA 95050
USA

Email: yingzhen.qu@huawei.com

Jeff Tantsura
Futurewei Technologies, Inc.
2330 Central Expressway
Santa Clara, CA 95050
USA

Email: jefftant.ietf@gmail.com

Network Working Group A. Clemm
Internet-Draft Futurewei Technologies, Inc.
Intended status: Informational E. Voit
Expires: April 21, 2018 Cisco Systems
X. Liu
Jabil
I. Bryskin
T. Zhou
G. Zheng
Huawei
H. Birkholz
Fraunhofer SIT
October 18, 2017

Smart filters for Push Updates - Problem Statement
draft-clemm-netconf-push-smart-filters-ps-00

Abstract

This document defines a problem statement for Smart Filters for Push Updates. Smart Filters for Push Updates (referred to simply as "Smart Filters" in the context of this document) allows to filter push updates based on values of pushed objects and/or state, such as previous updates. Smart Filters provide an important building block for service assurance and network automation.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

YANG-Push [yang-push] allows client applications to subscribe to continuous datastore updates without needing to poll. YANG-Push subscriptions allow client applications to select which datanodes are of interest. For this purpose, filters that act as node selectors are offered. However, what is currently not supported are filters that filter updates based on values, such as sending updates only when the value falls within a certain range. Also not supported are filters that would require additional state, such as sending updates only when the value exceeds a certain threshold for the first time but not again until the threshold is cleared. We refer to such filters as "smart filters", with further subcategories of "smart stateless filters" and "smart stateful filters", respectively.

Smart filters involve more complex subscription and implementation semantics than the simple selection filters that are currently offered as part of YANG-Push. They involve post processing of updates that goes beyond basic update generation for polling avoidance and place additional intelligence at the server. Because of this, smart filter functionality was not included in the YANG-Push specification, although it was recognized that YANG-Push could be extended to include such functionality if needed. This is the purpose of this specification.

Smart filters facilitate service assurance, because they allow client applications to focus on "outliers" and updates that signify exceptions and conditions of interest have the biggest operational significance. They save network resources by avoiding the need to stream updates that would be discarded anyway, and allow applications to scale better since larger networks imply a larger amount of smart filtering operations delegated away from the application to the network. Smart filters also facilitate network automation as they constitute an important ingredient to specify triggers for automated actions.

2. Key Words

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

3. Definitions and Acronyms

Smart Filter: A filter that involves some processing, such as comparing values or differentiating behavior depending on state.

TCA: Threshold Crossing Alert.

YANG-Push: A server capability that allows client applications to subscribe to network management datastore updates.

4. Problem Statement

YANG-Push provides client applications with the ability to subscribe to continuous updates from network management datastores, obviating the need to perform polling and resulting in more robust and efficient applications. However, many applications do not require every update, only updates that are of certain interest.

For example, an update concerning interface utilization may be only needed when a certain utilization level is breached. Sending continuous updates when utilization is low might divert processing resources away from updates regarding interfaces whose utilization level may reach a critical point that requires attention. Doing so will require a filter based on an object value. Even sending continuous updates when utilization is high may be too much and counterproductive. It may be sufficient to send an update when a threshold is breached to raise a flag of attention, but then not to continue sending updates while the condition still persists but simply let the client application know when the threshold is cleared. This behavior cannot be accomplished simply by a value-based filter, but requires additional state to be maintained (so that the server has a memory whether or not the condition of a breached threshold has already been reported in prior update cycles).

What is needed are "Smart Filters" that provide the ability to apply filters based on object values, possibly also state. Smart Filters are useful for Service Assurance applications that need to monitor operational data for values that fall outside normal operational ranges. They are also useful for network automation, in which automated actions are automatically triggered based on when certain events in the network occur while certain conditions hold. A YANG-Push subscription with a smart filter can in effect act as a source for such events. Combined with an optional check for a condition when an event is observed, this can serve as the basis of action triggers.

Of course, it is possible to conceive filters that are very smart and powerful yet also very complex. While filters as defined in YANG-Push may be a tad too simple for the applications envisioned here, it is important to keep filters still simple enough to ensure broad implementation and support by networking devices. The smart filters defined in this effort intend to apply the "90/10" rule, aiming at the sweet spot that addresses 90% of use cases and deployment scenarios that can be addressed using 10% of the complexity. Where those filters are not sufficient, additional filters can be introduced outside this document.

It is proposed that Smart Filters for Push Updates will provide support for the following features:

- Support for smart filter extensions to YANG-Push subscriptions. The targeted model takes a "base" YANG-Push subscription and subjects updates to an additional filtering stage that is based on an object's value.

  * Filters that match or compare the object value against a fixed term or expression.

  * Filters that match or compare an object value against the value of another object. (This feature is useful particularly in conjunction with possible extensions that allow to compute aggregates. Such an ability opens the possibility to compare an object's current value to its mean, for example.)

o Support for refined on-change update semantics that allow client to distinguish whether object values were omitted or included because the object was created or deleted, or because the object's value fell outside filter range.

o Support for selected stateful filters:

   * This includes specifically support for generalized "threshold crossing alert" filters, or filters that provide an update only when an object's value passes a filter for the first time, and not again until the object's value passes a counter filter. In effect, the support involves attaching filter and counter filter to an object, including a switch at the object indicating which filter is in effect, and providing a distinction in the update which filter (e.g. onset of clear) was applied.

   * It may include additional filters, such a "recent high water mark" filters that allow to specify a time horizon until the current high water mark clears. A recent high water mark filter sends an update to an object only if its new value is greater than the last value that had been previously reported.

In order to constrain complexity, it is proposed that the following items will be outside the scope, subject to discussion by the Working Group:

o Filters that involve freely programmable logic.

o Filters that aggregate or otherwise process information over time. An example would be filters that compute an aggregate over a time series of data, for example, an object's average or top percentile value. (One way in which this can be accomplished is by defining a separate YANG module that allows to specify aggregates independent of any filtering, then asking users to subscribe to updates of the aggregate objects and applying filters there. The definition of such an aggregation module goes beyond the scope of the work defined here.)

o Filters that aggregate object's values with those of other objects, such as the maximum or average from objects over a list, or that operate on a function of other objects. An example would be an object for interface utilization that gets computed from objects for interface speed and interface packet rate, with the packet rate object itself potentially computed from counter snapshots that are taken at different times. (One way in which this can be accomplished is by defining a separate YANG module that allows to define objects that compute such functions akin to the Expression MIB [RFC2982], then asking users to subscribe to updates of those objects and applying filters there.)

5. IANA Considerations

Not applicable

6. Security Considerations

The application of smart filters requires a certain amount of processing resources at the server. An attacker could attempt to attack a server by creating YANG-push subscriptions with a large number of complex smart filters in an attempt to diminish server resources. Server implementations can guard against such scenarios in several ways. For one, they can implement NACM [RFC6536] in order to require proper authorization for requests to be made. Second, server implementations can reject requests made for a a larger number of smart filters than the implementation can reasonably sustain.

7. Normative References

[notif-sub]
Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Custom subscription to event notifications", July 2017, <https://datatracker.ietf.org/doc/draft-ietf-netconf-subscribed-notifications/>.

[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>.

[RFC2982] Kavasseri, R., Ed., "Distributed Management Expression MIB", RFC 2982, DOI 10.17487/RFC2982, October 2000, <https://www.rfc-editor.org/info/rfc2982>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <https://www.rfc-editor.org/info/rfc6536>.

[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>.

[yang-push]
Clemm, A., Voit, E., Gonzalez Prieto, A., Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "Subscribing to YANG datastore push updates", August 2017, <https://datatracker.ietf.org/doc/draft-ietf-netconf-yang-push/>.

Authors' Addresses

Alexander Clemm
Futurewei Technologies, Inc.
2330 Central Expressway
Santa Clara, CA 95050
USA

Email: ludwig@clemm.org

Eric Voit
Cisco Systems

Email: evoit@cisco.com

Xufeng Liu
Jabil

Email: Xufeng_Liu@jabil.com

Igor Bryskin
Huawei

Email: Igor.Bryskin@huawei.com

Tianran Zhou
Huawei

Email: zhoutianran@huawei.com

Guangying Zheng
Huawei

Email: zhengguangying@huawei.com

Henk Birkholz
Fraunhofer SIT

Email: henk.birkholz@sit.fraunhofer.de

NETCONF Working Group K. Watsen
Internet-Draft Watsen Networks
Intended status: Standards Track July 2, 2019
Expires: January 3, 2020

A YANG Data Model for a Keystore
draft-ietf-netconf-keystore-12

Abstract

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

Editorial Note (To be removed by RFC Editor)

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

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

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

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

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

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

o Appendix A. Change Log

Status of This Memo

Table of Contents

1. Introduction

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

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

Special consideration has been given for systems that have cryptographic hardware, such as a Trusted Protection Module (TPM). These systems are unique in that the cryptographic hardware hides the secret key values. To support such hardware, symmetric keys may have the value "hidden-key" and asymmetric keys may have the value "hidden-private-key". While how such keys are created or destroyed is outside the scope of this document, the keystore can contain entries for such keys, enabling them to be reference by other configuration elements.

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

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

2. Requirements Language

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

3. The Keystore Model

3.1. Tree Diagram

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

```
module: ietf-keystore
  +--rw keystore
     +--rw asymmetric-keys
     |  +--rw asymmetric-key* [name]
     |     +--rw name                                    string
     |     +--rw algorithm
     |     |       asymmetric-key-algorithm-t
     |     +--rw public-key                              binary
     |     +--rw (private-key-type)
     |     |  +--:(private-key)
     |     |  |  +--rw private-key?                      binary
     |     |  +--:(hidden-private-key)
     |     |  |  +--rw hidden-private-key?               empty
     |     |  +--:(encrypted-private-key)
     |     |     +--rw encrypted-private-key
     |     |        +--rw (key-type)
     |     |        |  +--:(symmetric-key-ref)
     |     |        |  |  +--rw symmetric-key-ref?    leafref
     |     |        |  |          {keystore-supported}?
     |     |        |  +--:(asymmetric-key-ref)
     |     |        |     +--rw asymmetric-key-ref?   leafref
     |     |        |             {keystore-supported}?
     |     |        +--rw value?                      binary
     |     +--rw certificates
     |     |  +--rw certificate* [name]
     |     |     +--rw name                       string
     |     |     +--rw cert?                      end-entity-cert-cms
     |     |     +---n certificate-expiration
     |     |        +-- expiration-date    yang:date-and-time
     |     +---x generate-certificate-signing-request
     |        +---w input
     |        |  +---w subject       binary
     |        |  +---w attributes?   binary
     |        +--ro output
     |           +--ro certificate-signing-request    binary
     +--rw symmetric-keys
        +--rw symmetric-key* [name]
           +--rw name                  string
           +--rw algorithm             encryption-algorithm-t
           +--rw (key-type)
```

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

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

```
          |      +---w (key-type)
          |         +--:(symmetric-key-ref)
          |         |  +---w symmetric-key-ref?    leafref
          |         |          {keystore-supported}?
          |         +--:(asymmetric-key-ref)
          |            +---w asymmetric-key-ref?   leafref
          |                    {keystore-supported}?
          +--ro output
             +--ro algorithm
             |       asymmetric-key-algorithm-t
             +--ro public-key                    binary
             +--ro (private-key-type)
                +--:(private-key)
                |  +--ro private-key?            binary
                +--:(hidden-private-key)
                |  +--ro hidden-private-key?     empty
                +--:(encrypted-private-key)
                   +--ro encrypted-private-key
                      +--ro (key-type)
                      |  +--:(symmetric-key-ref)
                      |  |  +--ro symmetric-key-ref?    leafref
                      |  |          {keystore-supported}?
                      |  +--:(asymmetric-key-ref)
                      |     +--ro asymmetric-key-ref?   leafref
                      |             {keystore-supported}?
                      +--ro value?                      binary

     grouping key-reference-type-grouping
       +-- (key-type)
          +--:(symmetric-key-ref)
          |  +-- symmetric-key-ref?
          |          -> /keystore/symmetric-keys/symmetric-key/name
          |          {keystore-supported}?
          +--:(asymmetric-key-ref)
             +-- asymmetric-key-ref?
                     -> /keystore/asymmetric-keys/asymmetric-key/name
                     {keystore-supported}?
     grouping encrypted-value-grouping
       +-- (key-type)
       |  +--:(symmetric-key-ref)
       |  |  +-- symmetric-key-ref?
       |  |          -> /keystore/symmetric-keys/symmetric-key/name
       |  |          {keystore-supported}?
       |  +--:(asymmetric-key-ref)
       |     +-- asymmetric-key-ref?
       |             -> /keystore/asymmetric-keys/asymmetric-key/name
       |             {keystore-supported}?
       +-- value?                      binary
```

```
     grouping symmetric-key-grouping
       +-- algorithm             encryption-algorithm-t
       +-- (key-type)
          +--:(key)
          |  +-- key?            binary
          +--:(hidden-key)
          |  +-- hidden-key?     empty
          +--:(encrypted-key)
             +-- encrypted-key
                +-- (key-type)
                |  +--:(symmetric-key-ref)
                |  |  +-- symmetric-key-ref?    leafref
                |  |          {keystore-supported}?
                |  +--:(asymmetric-key-ref)
                |     +-- asymmetric-key-ref?   leafref
                |             {keystore-supported}?
                +-- value?                      binary
     grouping asymmetric-key-pair-grouping
       +-- algorithm                     asymmetric-key-algorithm-t
       +-- public-key                    binary
       +-- (private-key-type)
          +--:(private-key)
          |  +-- private-key?            binary
          +--:(hidden-private-key)
          |  +-- hidden-private-key?     empty
          +--:(encrypted-private-key)
             +-- encrypted-private-key
                +-- (key-type)
                |  +--:(symmetric-key-ref)
                |  |  +-- symmetric-key-ref?    leafref
                |  |          {keystore-supported}?
                |  +--:(asymmetric-key-ref)
                |     +-- asymmetric-key-ref?   leafref
                |             {keystore-supported}?
                +-- value?                      binary
     grouping asymmetric-key-pair-with-cert-grouping
       +-- algorithm
       |       asymmetric-key-algorithm-t
       +-- public-key                             binary
       +-- (private-key-type)
       |  +--:(private-key)
       |  |  +-- private-key?                     binary
       |  +--:(hidden-private-key)
       |  |  +-- hidden-private-key?              empty
       |  +--:(encrypted-private-key)
       |     +-- encrypted-private-key
       |        +-- (key-type)
       |        |  +--:(symmetric-key-ref)
```

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

```
       +-- certificate?      leafref
     grouping local-or-keystore-asymmetric-key-grouping
       +-- (local-or-keystore)
          +--:(local) {local-definitions-supported}?
          |  +-- local-definition
          |     +-- algorithm
          |     |       asymmetric-key-algorithm-t
          |     +-- public-key                     binary
          |     +-- (private-key-type)
          |        +--:(private-key)
          |        |  +-- private-key?             binary
          |        +--:(hidden-private-key)
          |        |  +-- hidden-private-key?      empty
          |        +--:(encrypted-private-key)
          |           +-- encrypted-private-key
          |              +-- (key-type)
          |              |  +--:(symmetric-key-ref)
          |              |  |  +-- symmetric-key-ref?    leafref
          |              |  |          {keystore-supported}?
          |              |  +--:(asymmetric-key-ref)
          |              |     +-- asymmetric-key-ref?   leafref
          |              |             {keystore-supported}?
          |              +-- value?                      binary
          +--:(keystore) {keystore-supported}?
             +-- keystore-reference?   ks:asymmetric-key-ref
     grouping local-or-keystore-asymmetric-key-with-certs-grouping
       +-- (local-or-keystore)
          +--:(local) {local-definitions-supported}?
          |  +-- local-definition
          |     +-- algorithm
          |     |       asymmetric-key-algorithm-t
          |     +-- public-key                              binary
          |     +-- (private-key-type)
          |     |  +--:(private-key)
          |     |  |  +-- private-key?                      binary
          |     |  +--:(hidden-private-key)
          |     |  |  +-- hidden-private-key?               empty
          |     |  +--:(encrypted-private-key)
          |     |     +-- encrypted-private-key
          |     |        +-- (key-type)
          |     |        |  +--:(symmetric-key-ref)
          |     |        |  |  +-- symmetric-key-ref?    leafref
          |     |        |  |          {keystore-supported}?
          |     |        |  +--:(asymmetric-key-ref)
          |     |        |     +-- asymmetric-key-ref?   leafref
          |     |        |             {keystore-supported}?
          |     |        +-- value?                      binary
          |     +-- certificates
```

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

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

```
                      |  |  +-- symmetric-key-ref?    leafref
                      |  |          {keystore-supported}?
                      |  +--:(asymmetric-key-ref)
                      |     +-- asymmetric-key-ref?   leafref
                      |             {keystore-supported}?
                      +-- value?                      binary
```

3.2. Example Usage

3.2.1. A Keystore Instance

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

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


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

  <!-- Asymmetric Keys -->
  <asymmetric-keys>

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

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

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

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

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

    </asymmetric-keys>

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

      <symmetric-key>
        <name>operators-encrypted-key</name>
        <algorithm>aes-256-cbc</algorithm>
        <encrypted-key>
          <asymmetric-key-ref>tpm-protected-key</asymmetric-key-ref>
          <value>base64encodedvalue==</value>
        </encrypted-key>
      </symmetric-key>

    </symmetric-keys>
```

```
</keystore>
```

3.2.2. The "generate-symmetric-key" RPC

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

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

Following is the complimentary RPC-reply.

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

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

3.2.3. Notable Keystore Groupings

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

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

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

  import ietf-keystore {
```

```
       prefix ks;
       reference
         "RFC VVVV: YANG Data Model for a 'Keystore' Mechanism";
     }

     organization
      "Example Corporation";

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

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

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

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

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

       list key-with-certs {
         key name;
         leaf name {
           type string;
           description
             "An arbitrary name for this key.";
```

```
         }
         uses ks:local-or-keystore-asymmetric-key-with-certs-grouping;
         description
           "An asymmetric key and its associated certs, that may be
            configured locally or be a reference to an asymmetric key
            (and its associated certs) in the keystore.";
       }

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

   }
```

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

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

<keystore-usage xmlns="http://example.com/ns/example-keystore-usage">

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

  <just-a-key>
    <name>a locally-defined key</name>
    <local-definition>
      <algorithm>rsa2048</algorithm>
      <private-key>base64encodedvalue==</private-key>
      <public-key>base64encodedvalue==</public-key>
    </local-definition>
  </just-a-key>

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

```
        <keystore-reference>ex-rsa-key</keystore-reference>
      </just-a-key>

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

      <key-with-certs>
        <name>a locally-defined key with certs</name>
        <local-definition>
          <algorithm>rsa2048</algorithm>
          <private-key>base64encodedvalue==</private-key>
          <public-key>base64encodedvalue==</public-key>
          <certificates>
            <certificate>
              <name>a locally-defined cert</name>
              <cert>base64encodedvalue==</cert>
            </certificate>
          </certificates>
        </local-definition>
      </key-with-certs>

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

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

      <end-entity-cert-with-key>
        <name>a locally-defined end-entity cert with key</name>
        <local-definition>
          <algorithm>rsa2048</algorithm>
          <private-key>base64encodedvalue==</private-key>
          <public-key>base64encodedvalue==</public-key>
          <cert>base64encodedvalue==</cert>
        </local-definition>
      </end-entity-cert-with-key>

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

    </keystore-usage>
```

3.3. YANG Module

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

```
<CODE BEGINS> file "ietf-keystore@2019-07-02.yang"

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

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

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

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

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

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

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

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

     This version of this YANG module is part of RFC XXXX
```

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

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

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

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

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

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

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

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

     typedef asymmetric-key-ref {
       type leafref {
         path "/ks:keystore/ks:asymmetric-keys/ks:asymmetric-key"
```

```
             + "/ks:name";
        }
        description
          "This typedef enables modules to easily define a reference
           to an asymmetric key stored in the keystore.";
      }

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


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

      grouping encrypted-value-grouping {
        description
          "A reusable grouping for a value that has been encrypted by
           a symmetric or asymmetric key in the keystore.";
        uses "key-reference-type-grouping";
        leaf value {
```

```
         type binary;
         description
           "The private key, encrypted using the specified symmetric
            or asymmetric key.";
       }
     }

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

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

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

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

     grouping asymmetric-key-certificate-ref-grouping {
       leaf asymmetric-key {
         type ks:asymmetric-key-ref;
         must '../certificate';
         description
           "A reference to an asymmetric key in the keystore.";
```

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

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

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

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

```
           if-feature "keystore-supported";
           container keystore-reference {
             uses asymmetric-key-certificate-ref-grouping;
             description
               "A reference to a specific certificate (and its
                associated private key) in the keystore.";
           }
         }
         description
           "A choice between an inlined definition and a definition
            that exists in the keystore.";
       }
     }

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

```
        }
      }
    } // grouping keystore-grouping


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

    container keystore {
      nacm:default-deny-write;
      description
        "The keystore contains a list of keys.";
      uses keystore-grouping;
    }

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

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

   }

   <CODE ENDS>
```

4. Security Considerations

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

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

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

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

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

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

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

5. IANA Considerations

5.1. The IETF XML Registry

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

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

5.2. The YANG Module Names Registry

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

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

6. References

6.1. Normative References

[I-D.ietf-netconf-crypto-types]
Watsen, K. and H. Wang, "Common YANG Data Types for Cryptography", draft-ietf-netconf-crypto-types-09 (work in progress), June 2019.

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

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

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

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

6.2. Informative References

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

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

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

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

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

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

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

Appendix A. Change Log

A.1. 00 to 01

- o Replaced the 'certificate-chain' structures with PKCS#7 structures. (Issue #1)

- o Added 'private-key' as a configurable data node, and removed the 'generate-private-key' and 'load-private-key' actions. (Issue #2)

- o Moved 'user-auth-credentials' to the ietf-ssh-client module. (Issues #4 and #5)

A.2. 01 to 02

- o Added back 'generate-private-key' action.

- o Removed 'RESTRICTED' enum from the 'private-key' leaf type.

- o Fixed up a few description statements.

A.3. 02 to 03

- o Changed draft's title.

- o Added missing references.

- o Collapsed sections and levels.

- o Added RFC 8174 to Requirements Language Section.

- o Renamed 'trusted-certificates' to 'pinned-certificates'.

- o Changed 'public-key' from config false to config true.

- o Switched 'host-key' from OneAsymmetricKey to definition from RFC 4253.

A.4. 03 to 04

- o Added typedefs around leafrefs to common keystore paths

- o Now tree diagrams reference ietf-netmod-yang-tree-diagrams

- o Removed Design Considerations section

- o Moved key and certificate definitions from data tree to groupings

A.5. 04 to 05

- 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 posibility 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 descendent 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.

o Added examples for the 'generate-symmetric-key' and 'generate-asymmetric-key' RPCs.

o Updated the Introduction section.

Acknowledgements

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

Author's Address

Kent Watsen
Watsen Networks

EMail: kent+ietf@watsen.net

NETCONF Working Group   K. Watsen
Internet-Draft   Watsen Networks
Intended status: Standards Track   July 2, 2019
Expires: January 3, 2020

NETCONF Client and Server Models
draft-ietf-netconf-netconf-client-server-14

Abstract

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

Editorial Note (To be removed by RFC Editor)

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

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

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

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

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

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

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

o Appendix B. Change Log

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

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

2. Terminology

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

3. The NETCONF Client Model

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

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

3.1. Tree Diagram

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

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

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

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

  grouping netconf-client-grouping
    +-- initiate! {ssh-initiate or tls-initiate}?
    |  +-- netconf-server* [name]
    |     +-- name?                 string
    |     +-- endpoints
    |     |  +-- endpoint* [name]
    |     |     +-- name?        string
    |     |     +-- (transport)
    |     |        +--:(ssh) {ssh-initiate}?
    |     |        |  +-- ssh
    |     |        |     +-- tcp-client-parameters
    |     |        |     |  +---u tcpc:tcp-client-grouping
    |     |        |     +-- ssh-client-parameters
```

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

3.2. Example Usage

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

This example is consistent with the examples presented in Section 2 of [I-D.ietf-netconf-trust-anchors] and Section 3.2 of [I-D.ietf-netconf-keystore].

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

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

  <!-- NETCONF servers to initiate connections to -->
  <initiate>
    <netconf-server>
      <name>corp-fw1</name>
      <endpoints>
        <endpoint>
          <name>corp-fw1.example.com</name>
          <ssh>
            <tcp-client-parameters>
              <remote-address>corp-fw1.example.com</remote-address>
              <keepalives>
                <idle-time>15</idle-time>
                <max-probes>3</max-probes>
                <probe-interval>30</probe-interval>
              </keepalives>
            </tcp-client-parameters>
            <ssh-client-parameters>
              <client-identity>
                <username>foobar</username>
                <public-key>
                  <local-definition>
                    <algorithm>rsa2048</algorithm>
                    <private-key>base64encodedvalue==</private-key>
                    <public-key>base64encodedvalue==</public-key>
                  </local-definition>
                </public-key>
              </client-identity>
              <server-authentication>
                <ca-certs>explicitly-trusted-server-ca-certs</ca-cer\
ts>
                <server-certs>explicitly-trusted-server-certs</serve\
r-certs>
              </server-authentication>
              <keepalives>
                <max-wait>30</max-wait>
                <max-attempts>3</max-attempts>
              </keepalives>
            </ssh-client-parameters>
          </ssh>
        </endpoint>
        <endpoint>
          <name>corp-fw2.example.com</name>
          <ssh>
```

```
               <tcp-client-parameters>
                 <remote-address>corp-fw2.example.com</remote-address>
                 <keepalives>
                   <idle-time>15</idle-time>
                   <max-probes>3</max-probes>
                   <probe-interval>30</probe-interval>
                 </keepalives>
               </tcp-client-parameters>
               <ssh-client-parameters>
                 <client-identity>
                   <username>foobar</username>
                   <public-key>
                     <local-definition>
                       <algorithm>rsa2048</algorithm>
                       <private-key>base64encodedvalue==</private-key>
                       <public-key>base64encodedvalue==</public-key>
                     </local-definition>
                   </public-key>
                 </client-identity>
                 <server-authentication>
                   <ca-certs>explicitly-trusted-server-ca-certs</ca-cer\
ts>
                   <server-certs>explicitly-trusted-server-certs</serve\
r-certs>
                 </server-authentication>
                 <keepalives>
                   <max-wait>30</max-wait>
                   <max-attempts>3</max-attempts>
                 </keepalives>
               </ssh-client-parameters>
             </ssh>
           </endpoint>
         </endpoints>
         <connection-type>
           <persistent/>
         </connection-type>
         <reconnect-strategy>
           <start-with>last-connected</start-with>
         </reconnect-strategy>
       </netconf-server>
     </initiate>

     <!-- endpoints to listen for NETCONF Call Home connections on -->
     <listen>
       <endpoint>
         <name>Intranet-facing listener</name>
         <ssh>
           <tcp-server-parameters>
```

```
            <local-address>192.0.2.7</local-address>
          </tcp-server-parameters>
          <ssh-client-parameters>
            <client-identity>
              <username>foobar</username>
              <public-key>
                <local-definition>
                  <algorithm>rsa2048</algorithm>
                  <private-key>base64encodedvalue==</private-key>
                  <public-key>base64encodedvalue==</public-key>
                </local-definition>
              </public-key>
            </client-identity>
            <server-authentication>
              <ca-certs>explicitly-trusted-server-ca-certs</ca-certs>
              <server-certs>explicitly-trusted-server-certs</server-ce\
rts>
              <ssh-host-keys>explicitly-trusted-ssh-host-keys</ssh-hos\
t-keys>
            </server-authentication>
          </ssh-client-parameters>
        </ssh>
      </endpoint>
    </listen>
  </netconf-client>
```

3.3. YANG Module

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

```
<CODE BEGINS> file "ietf-netconf-client@2019-07-02.yang"
module ietf-netconf-client {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-netconf-client";
  prefix ncc;

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

  import ietf-tcp-client {
    prefix tcpc;
    reference
```

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

    import ietf-ssh-client {
      prefix sshc;
      revision-date 2019-07-02; // stable grouping definitions
      reference
        "RFC BBBB: YANG Groupings for SSH Clients and SSH Servers";
    }

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

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

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

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

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

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

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

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

   revision 2019-07-02 {
     description
       "Initial version";
     reference
       "RFC XXXX: 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
         "Top-level grouping for NETCONF client configuration.";
       container initiate {
         if-feature "ssh-initiate or tls-initiate";
         presence "Enables client to initiate TCP connections";
         description
           "Configures client initiating underlying TCP connections.";
         list netconf-server {
           key "name";
           min-elements 1;
           description
             "List of NETCONF servers the NETCONF client is to
              initiate connections to in parallel.";
           leaf name {
             type string;
             description
               "An arbitrary name for the NETCONF server.";
           }
           container endpoints {
             description
               "Container for the list of endpoints.";
             list endpoint {
               key "name";
               min-elements 1;
               ordered-by user;
               description
                 "A user-ordered list of endpoints that the NETCONF
                  client will attempt to connect to in the specified
                  sequence.  Defining more than one enables
                  high-availability.";
               leaf name {
                 type string;
                 description
                   "An arbitrary name for the endpoint.";
               }
               choice transport {
                 mandatory true;
                 description
                   "Selects between available transports.";
                 case ssh {
                   if-feature "ssh-initiate";
```

```
                   container ssh {
                     description
                       "Specifies IP and SSH specific configuration
                        for the connection.";
                     container tcp-client-parameters {
                       description
                         "A wrapper around the TCP client parameters
                          to avoid name collisions.";
                       uses tcpc:tcp-client-grouping {
                         refine "remote-port" {
                           default "830";
                           description
                             "The NETCONF client will attempt to connect
                              to the IANA-assigned well-known port value
                              for 'netconf-ssh' (443) if no value is
                              specified.";
                         }
                       }
                     }
                     container ssh-client-parameters {
                       description
                         "A wrapper around the SSH client parameters to
                          avoid name collisions.";
                       uses sshc:ssh-client-grouping;
                     }
                   }
                 }
                 case tls {
                   if-feature "tls-initiate";
                   container tls {
                     description
                       "Specifies IP and TLS specific configuration
                        for the connection.";
                     container tcp-client-parameters {
                       description
                         "A wrapper around the TCP client parameters
                          to avoid name collisions.";
                       uses tcpc:tcp-client-grouping {
                         refine "remote-port" {
                           default "6513";
                           description
                             "The NETCONF client will attempt to connect
                              to the IANA-assigned well-known port value
                              for 'netconf-tls' (6513) if no value is
                              specified.";
                         }
                       }
                     }
```

```
                  container tls-client-parameters {
                    must "client-identity" {
                      description
                        "NETCONF/TLS clients MUST pass some
                         authentication credentials.";
                    }
                    description
                      "A wrapper around the TLS client parameters
                       to avoid name collisions.";
                    uses tlsc:tls-client-grouping;
                  }
                }
              }
            } // choice transport
          } // list endpoint
        } // container endpoints

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

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

                   This connection type increases resource
```

```
                      utilization, albeit with increased delay in
                      NETCONF server to NETCONF client interactions.

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

                      In the case that the previous connection is still
                      active, establishing a new connection is NOT
                      RECOMMENDED.";
                   leaf period {
                     type uint16;
                     units "minutes";
                     default "60";
                     description
                       "Duration of time between periodic connections.";
                   }
                   leaf anchor-time {
                     type yang:date-and-time {
                       // constrained to minute-level granularity
                       pattern '\d{4}-\d{2}-\d{2}T\d{2}:\d{2}'
                             + '(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 "Enables client to accept call-home connections";
          description
            "Configures client accepting call-home TCP connections.";
          leaf idle-timeout {
            type uint16;
            units "seconds";
            default "3600"; // one hour
            description
              "Specifies the maximum number of seconds that a NETCONF
               session may remain idle. A NETCONF session will be
               dropped if it is idle for an interval longer than this
               number of seconds.  If set to zero, then the server
               will never drop a session because it is idle.  Sessions
               that have a notification subscription active are never
               dropped.";
          }
          list endpoint {
            key "name";
            min-elements 1;
            description
              "List of endpoints to listen for NETCONF connections.";
            leaf name {
              type string;
              description
                "An arbitrary name for the NETCONF listen endpoint.";
            }
            choice transport {
              mandatory true;
              description
                "Selects between available transports.";
              case ssh {
                if-feature "ssh-listen";
                container ssh {
                  description
                    "SSH-specific listening configuration for inbound
                     connections.";
                  container tcp-server-parameters {
                    description
                      "A wrapper around the TCP server parameters
                       to avoid name collisions.";
                    uses tcps:tcp-server-grouping {
                      refine "local-port" {
                        default "4334";
```

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

```
           } // transport
         } // endpoint
       } // listen
     } // netconf-client

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

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

4. The NETCONF Server Model

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

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

4.1. Tree Diagram

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

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

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

```
   module: ietf-netconf-server
     +--rw netconf-server
        +---u netconf-server-grouping

     grouping netconf-server-grouping
       +-- listen! {ssh-listen or tls-listen}?
       |  +-- idle-timeout?   uint16
       |  +-- endpoint* [name]
       |     +-- name?        string
       |     +-- (transport)
```

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

4.2. Example Usage

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

This example is consistent with the examples presented in Section 2 of [I-D.ietf-netconf-trust-anchors] and Section 3.2 of [I-D.ietf-netconf-keystore].

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


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

  <!-- endpoints to listen for NETCONF connections on -->
  <listen>
    <endpoint> <!-- listening for SSH connections -->
      <name>netconf/ssh</name>
      <ssh>
        <tcp-server-parameters>
          <local-address>192.0.2.7</local-address>
        </tcp-server-parameters>
        <ssh-server-parameters>
          <server-identity>
            <host-key>
              <name>deployment-specific-certificate</name>
              <public-key>
                <local-definition>
                  <algorithm>rsa2048</algorithm>
                  <private-key>base64encodedvalue==</private-key>
                  <public-key>base64encodedvalue==</public-key>
                </local-definition>
              </public-key>
            </host-key>
          </server-identity>
          <client-authentication>
            <supported-authentication-methods>
              <publickey/>
            </supported-authentication-methods>
            <client-auth-defined-elsewhere/>
          </client-authentication>
        </ssh-server-parameters>
      </ssh>
    </endpoint>
    <endpoint> <!-- listening for TLS sessions -->
```

```
          <name>netconf/tls</name>
          <tls>
            <tcp-server-parameters>
              <local-address>192.0.2.7</local-address>
            </tcp-server-parameters>
            <tls-server-parameters>
              <server-identity>
                <local-definition>
                  <algorithm>rsa2048</algorithm>
                  <private-key>base64encodedvalue==</private-key>
                  <public-key>base64encodedvalue==</public-key>
                  <cert>base64encodedvalue==</cert>
                </local-definition>
              </server-identity>
              <client-authentication>
                <required/>
                <ca-certs>explicitly-trusted-client-ca-certs</ca-certs>
                <client-certs>explicitly-trusted-client-certs</client-ce\
   rts>
                <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-authentication>
            </tls-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>
            </tcp-client-parameters>
            <ssh-server-parameters>
              <server-identity>
                <host-key>
                  <name>deployment-specific-certificate</name>
                  <public-key>
                    <local-definition>
                      <algorithm>rsa2048</algorithm>
                      <private-key>base64encodedvalue==</private-key>
                      <public-key>base64encodedvalue==</public-key>
                    </local-definition>
                  </public-key>
                </host-key>
              </server-identity>
              <client-authentication>
                <supported-authentication-methods>
                  <publickey/>
                </supported-authentication-methods>
                <client-auth-defined-elsewhere/>
              </client-authentication>
            </ssh-server-parameters>
          </ssh>
        </endpoint>
        <endpoint>
          <name>west-data-center</name>
          <ssh>
            <tcp-client-parameters>
              <remote-address>west.config-mgr.example.com</remote-ad\
dress>
            </tcp-client-parameters>
            <ssh-server-parameters>
              <server-identity>
                <host-key>
                  <name>deployment-specific-certificate</name>
                  <public-key>
                    <local-definition>
                      <algorithm>rsa2048</algorithm>
                      <private-key>base64encodedvalue==</private-key>
                      <public-key>base64encodedvalue==</public-key>
                    </local-definition>
                  </public-key>
                </host-key>
              </server-identity>
              <client-authentication>
                <supported-authentication-methods>
                  <publickey/>
                </supported-authentication-methods>
```

```
                 <client-auth-defined-elsewhere/>
               </client-authentication>
             </ssh-server-parameters>
           </ssh>
         </endpoint>
       </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>
                 <local-definition>
                   <algorithm>rsa2048</algorithm>
                   <private-key>base64encodedvalue==</private-key>
                   <public-key>base64encodedvalue==</public-key>
                   <cert>base64encodedvalue==</cert>
                 </local-definition>
               </server-identity>
               <client-authentication>
                 <required/>
                 <ca-certs>explicitly-trusted-client-ca-certs</ca-cer\
ts>
                 <client-certs>explicitly-trusted-client-certs</clien\
t-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-authentication>
                <keepalives>
                  <max-wait>30</max-wait>
                  <max-attempts>3</max-attempts>
                </keepalives>
              </tls-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>
                  <local-definition>
                    <algorithm>rsa2048</algorithm>
                    <private-key>base64encodedvalue==</private-key>
                    <public-key>base64encodedvalue==</public-key>
                    <cert>base64encodedvalue==</cert>
                  </local-definition>
                </server-identity>
                <client-authentication>
                  <required/>
                  <ca-certs>explicitly-trusted-client-ca-certs</ca-cer\
   ts>
                  <client-certs>explicitly-trusted-client-certs</clien\
   t-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-authentication>
                <keepalives>
                  <max-wait>30</max-wait>
                  <max-attempts>3</max-attempts>
                </keepalives>
              </tls-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>
```

4.3. YANG Module

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

```
<CODE BEGINS> file "ietf-netconf-server@2019-07-02.yang"
module ietf-netconf-server {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-netconf-server";
  prefix ncs;

  import ietf-yang-types {
    prefix yang;
```

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

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

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

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

     import ietf-ssh-server {
       prefix sshs;
       revision-date 2019-07-02; // stable grouping definitions
       reference
         "RFC BBBB: YANG Groupings for SSH Clients and SSH Servers";
     }

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

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

     contact
       "WG Web:   <http://datatracker.ietf.org/wg/netconf/>
        WG List:  <mailto:netconf@ietf.org>
        Author:   Kent Watsen <mailto:kent+ietf@watsen.net>
        Author:   Gary Wu <mailto:garywu@cisco.com>
        Author:   Juergen Schoenwaelder
                  <mailto:j.schoenwaelder@jacobs-university.de>";
     description
       "This module contains a collection of YANG definitions
```

```
        for configuring NETCONF servers.

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

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

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

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

     revision 2019-07-02 {
       description
         "Initial version";
       reference
         "RFC XXXX: 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
         "Top-level grouping for NETCONF server configuration.";
       container listen {
         if-feature "ssh-listen or tls-listen";
         presence
           "Enables server to listen for NETCONF client connections.";
         description
           "Configures listen behavior";
         leaf idle-timeout {
           type uint16;
           units "seconds";
           default 3600; // one hour
           description
             "Specifies the maximum number of seconds that a NETCONF
              session may remain idle. A NETCONF session will be
              dropped if it is idle for an interval longer than this
              number of seconds.  If set to zero, then the server
              will never drop a session because it is idle.  Sessions
              that have a notification subscription active are never
              dropped.";
         }
         list endpoint {
```

```
          key "name";
          min-elements 1;
          description
            "List of endpoints to listen for NETCONF connections.";
          leaf name {
            type string;
            description
              "An arbitrary name for the NETCONF listen endpoint.";
          }
          choice transport {
            mandatory true;
            description
              "Selects between available transports.";
            case ssh {
              if-feature "ssh-listen";
              container ssh {
                description
                  "SSH-specific listening configuration for inbound
                   connections.";
                container tcp-server-parameters {
                  description
                    "A wrapper around the TCP client parameters
                     to avoid name collisions.";
                  uses tcps:tcp-server-grouping {
                    refine "local-port" {
                      default "830";
                      description
                        "The NETCONF server will listen on the
                         IANA-assigned well-known port value
                         for 'netconf-ssh' (830) if no value
                         is specified.";
                    }
                  }
                }
                container ssh-server-parameters {
                  description
                    "A wrapper around the SSH server parameters
                     to avoid name collisions.";
                  uses sshs:ssh-server-grouping;
                }
              }
            }
            case tls {
              if-feature "tls-listen";
              container tls {
                description
                  "TLS-specific listening configuration for inbound
                   connections.";
```

```
                 container tcp-server-parameters {
                   description
                     "A wrapper around the TCP client parameters
                      to avoid name collisions.";
                   uses tcps:tcp-server-grouping {
                     refine "local-port" {
                       default "6513";
                       description
                         "The NETCONF server will listen on the
                          IANA-assigned well-known port value
                          for 'netconf-tls' (6513) if no value
                          is specified.";
                     }
                   }
                 }
                 container tls-server-parameters {
                   description
                     "A wrapper around the TLS server parameters to
                      avoid name collisions.";
                   uses tlss:tls-server-grouping {
                     refine "client-authentication" {
                       //must 'ca-certs or client-certs';
                       description
                         "NETCONF/TLS servers MUST validate client
                          certificates.";
                     }
                     augment "client-authentication" {
                       description
                         "Augments in the cert-to-name structure.";
                       container cert-maps {
                         uses x509c2n:cert-to-name;
                         description
                           "The cert-maps container is used by a TLS-
                            based NETCONF server to map the NETCONF
                            client's presented X.509 certificate to
                            a NETCONF username.  If no matching and
                            valid cert-to-name list entry can be found,
                            then the NETCONF server MUST close the
                            connection, and MUST NOT accept NETCONF
                            messages over it.";
                         reference
                           "RFC WWWW: NETCONF over TLS, Section 7";
                       }
                     }
                   }
                 }
               }
             }
```

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

```
                         "A wrapper around the TCP client parameters
                          to avoid name collisions.";
                       uses tcpc:tcp-client-grouping {
                         refine "remote-port" {
                           default "4334";
                           description
                             "The NETCONF server will attempt to connect
                              to the IANA-assigned well-known port for
                              'netconf-ch-tls' (4334) if no value is
                              specified.";
                         }
                       }
                     }
                     container ssh-server-parameters {
                       description
                         "A wrapper around the SSH server parameters
                          to avoid name collisions.";
                       uses sshs:ssh-server-grouping;
                     }
                   }
                 }
                 case tls {
                   if-feature "tls-call-home";
                   container tls {
                     description
                       "Specifies TLS-specific call-home transport
                        configuration.";
                     container tcp-client-parameters {
                       description
                         "A wrapper around the TCP client parameters
                          to avoid name collisions.";
                       uses tcpc:tcp-client-grouping {
                         refine "remote-port" {
                           default "4335";
                           description
                             "The NETCONF server will attempt to connect
                              to the IANA-assigned well-known port for
                              'netconf-ch-tls' (4335) if no value is
                              specified.";
                         }
                       }
                     }
                     container tls-server-parameters {
                       description
                         "A wrapper around the TLS server parameters
                          to avoid name collisions.";
                       uses tlss:tls-server-grouping {
                         refine "client-authentication" {
```

```
                          /* commented out since auth could be external
                        must 'ca-certs or client-certs';
                        */
                          description
                            "NETCONF/TLS servers MUST validate client
                             certificates.";
                        }
                        augment "client-authentication" {
                          description
                            "Augments in the cert-to-name structure.";
                          container cert-maps {
                            uses x509c2n:cert-to-name;
                            description
                              "The cert-maps container is used by a
                               TLS-based NETCONF server to map the
                               NETCONF client's presented X.509
                               certificate to a NETCONF username.  If
                               no matching and valid cert-to-name list
                               entry can be found, then the NETCONF
                               server MUST close the connection, and
                               MUST NOT accept NETCONF messages over
                               it.";
                            reference
                              "RFC WWWW: NETCONF over TLS, Section 7";
                          }
                        }
                      }
                    }
                  }
                } // tls
              } // choice
            } // endpoint
          } // endpoints
          container connection-type {
            description
              "Indicates the NETCONF server's preference for how the
               NETCONF connection is maintained.";
            choice connection-type {
              mandatory true;
              description
                "Selects between available connection types.";
              case persistent-connection {
                container persistent {
                  presence "Indicates that a persistent connection is
                            to be maintained.";
                  description
                    "Maintain a persistent connection to the NETCONF
                     client. If the connection goes down, immediately
```

```
                      start trying to reconnect to the NETCONF client,
                      using the reconnection strategy.

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

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

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

                      In the case that the previous connection is still
                      active (i.e., the NETCONF client has not closed
                      it yet), establishing a new connection is NOT
                      RECOMMENDED.";
                   leaf period {
                     type uint16;
                     units "minutes";
                     default "60";
                     description
                       "Duration of time between periodic connections.";
                   }
                   leaf anchor-time {
                     type yang:date-and-time {
                       // constrained to minute-level granularity
                       pattern '\d{4}-\d{2}-\d{2}T\d{2}:\d{2}'
                             + '(Z|[\+\-]\d{2}:\d{2})';
                     }
                     description
                       "Designates a timestamp before or after which a
                        series of periodic connections are determined.
                        The periodic connections occur at a whole
                        multiple interval from the anchor time.  For
                        example, for an anchor time is 15 minutes past
```

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

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

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

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

5. Security Considerations

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

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

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

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

None of the subtrees or data nodes in the modules defined in this document need to be protected from write operations.

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

None of the subtrees or data nodes in the modules defined in this document need to be protected from read operations.

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

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

6. IANA Considerations

6.1. The IETF XML Registry

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

      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 [RFC6020]. Following the format in [RFC6020], the the following registrations are requested:

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

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

7. References

7.1. Normative References

[I-D.ietf-netconf-keystore]
Watsen, K., "A YANG Data Model for a Keystore", draft-ietf-netconf-keystore-11 (work in progress), June 2019.

[I-D.ietf-netconf-ssh-client-server]
Watsen, K., Wu, G., and L. Xia, "YANG Groupings for SSH Clients and SSH Servers", draft-ietf-netconf-ssh-client-server-14 (work in progress), June 2019.

[I-D.ietf-netconf-tls-client-server]
Watsen, K., Wu, G., and L. Xia, "YANG Groupings for TLS Clients and TLS Servers", draft-ietf-netconf-tls-client-server-13 (work in progress), June 2019.

[I-D.kwatsen-netconf-tcp-client-server]
Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", draft-kwatsen-netconf-tcp-client-server-02 (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>.

[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>.

7.2. Informative References

[I-D.ietf-netconf-trust-anchors]
Watsen, K., "A YANG Data Model for a Truststore", draft-ietf-netconf-trust-anchors-05 (work in progress), June 2019.

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

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

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

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

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

Appendix A. Expanded Tree Diagrams

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

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

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

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

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

module: ietf-netconf-client
  +--rw netconf-client
     +--rw initiate! {ssh-initiate or tls-initiate}?
     |  +--rw netconf-server* [name]
     |     +--rw name                  string
     |     +--rw endpoints
     |     |  +--rw endpoint* [name]
     |     |     +--rw name         string
     |     |     +--rw (transport)
     |     |        +--:(ssh) {ssh-initiate}?
     |     |        |  +--rw ssh
     |     |        |     +--rw tcp-client-parameters
     |     |        |     |  +--rw remote-address    inet:host
     |     |        |     |  +--rw remote-port?      inet:port-number
     |     |        |     |  +--rw local-address?    inet:ip-address
     |     |        |     |  |       {local-binding-supported}?
     |     |        |     |  +--rw local-port?       inet:port-number
     |     |        |     |  |       {local-binding-supported}?
     |     |        |     |  +--rw keepalives!
     |     |        |     |          {keepalives-supported}?
     |     |        |     |     +--rw idle-time         uint16
     |     |        |     |     +--rw max-probes        uint16
     |     |        |     |     +--rw probe-interval    uint16
     |     |        |     +--rw ssh-client-parameters
     |     |        |        +--rw client-identity
     |     |        |        |  +--rw username?            string
     |     |        |        |  +--rw (auth-type)
     |     |        |        |     +--:(password)
     |     |        |        |     |  +--rw password?      string
     |     |        |        |     +--:(public-key)
     |     |        |        |     |  +--rw public-key
     |     |        |        |     |     +--rw (local-or-keystore)
     |     |        |        |     |        +--:(local)
```

```
     |     |        |        |     |        |        {local-definiti\
\ons-supported}?
     |     |        |        |     |        |  +--rw local-definition
     |     |        |        |     |        |     +--rw algorithm
     |     |        |        |     |        |     |       asymmetric\
\-key-algorithm-t
     |     |        |        |     |        |     +--rw public-key
     |     |        |        |     |        |     |       binary
     |     |        |        |     |        |     +--rw (private-key\
\-type)
     |     |        |        |     |        |        +--:(private-ke\
\y)
     |     |        |        |     |        |        |  +--rw privat\
\e-key?
     |     |        |        |     |        |        |          bina\
\ry
     |     |        |        |     |        |        +--:(hidden-pri\
\vate-key)
     |     |        |        |     |        |        |  +--rw hidden\
\-private-key?
     |     |        |        |     |        |        |          empty
     |     |        |        |     |        |        +--:(encrypted-\
\private-key)
     |     |        |        |     |        |           +--rw encryp\
\ted-private-key
     |     |        |        |     |        |              +--rw (ke\
\y-type)
     |     |        |        |     |        |              |  +--:(s\
\ymmetric-key-ref)
     |     |        |        |     |        |              |  |  +--\
\rw symmetric-key-ref?    leafref
     |     |        |        |     |        |              |  |     \
\     {keystore-supported}?
     |     |        |        |     |        |              |  +--:(a\
\symmetric-key-ref)
     |     |        |        |     |        |              |     +--\
\rw asymmetric-key-ref?   leafref
     |     |        |        |     |        |              |        \
\     {keystore-supported}?
     |     |        |        |     |        |              +--rw val\
\ue?
     |     |        |        |     |        |                      b\
\inary
     |     |        |        |     |     +--:(keystore)
     |     |        |        |     |              {keystore-suppo\
\rted}?
     |     |        |        |     |        +--rw keystore-refere\
\nce?
```

```
     |     |       |       |    |                   ks:asymmetric\
\-key-ref
     |     |       |       |    +--:(certificate)
     |     |       |       |       +--rw certificate
     |     |       |       |               {sshcmn:ssh-x509-certs\
\}?
     |     |       |       |          +--rw (local-or-keystore)
     |     |       |       |             +--:(local)
     |     |       |       |             |         {local-definiti\
\ons-supported}?
     |     |       |       |             |  +--rw local-definition
     |     |       |       |             |     +--rw algorithm
     |     |       |       |             |     |       asymmetric\
\-key-algorithm-t
     |     |       |       |             |     +--rw public-key
     |     |       |       |             |     |       binary
     |     |       |       |             |     +--rw (private-key\
\-type)
     |     |       |       |             |     |  +--:(private-ke\
\y)
     |     |       |       |             |     |  |  +--rw privat\
\e-key?
     |     |       |       |             |     |  |          bina\
\ry
     |     |       |       |             |     |  +--:(hidden-pri\
\vate-key)
     |     |       |       |             |     |  |  +--rw hidden\
\-private-key?
     |     |       |       |             |     |  |          empty
     |     |       |       |             |     |  +--:(encrypted-\
\private-key)
     |     |       |       |             |     |     +--rw encryp\
\ted-private-key
     |     |       |       |             |     |        +--rw (ke\
\y-type)
     |     |       |       |             |     |        |  +--:(s\
\ymmetric-key-ref)
     |     |       |       |             |     |        |  |  +--\
\rw symmetric-key-ref?    leafref
     |     |       |       |             |     |        |  |     \
\     {keystore-supported}?
     |     |       |       |             |     |        |  +--:(a\
\symmetric-key-ref)
     |     |       |       |             |     |        |     +--\
\rw asymmetric-key-ref?   leafref
     |     |       |       |             |     |        |        \
\     {keystore-supported}?
     |     |       |       |             |     |        +--rw val\
```

```
   \ue?
        |     |        |        |              |     |               b\
   \inary
        |     |        |        |              |     +--rw cert?
        |     |        |        |              |     |       end-entity\
   \-cert-cms
        |     |        |        |              |     +---n certificate-\
   \expiration
        |     |        |        |              |     |  +-- expiration-\
   \date
        |     |        |        |              |     |          yang:da\
   \te-and-time
        |     |        |        |              |     +---x generate-cer\
   \tificate-signing-request
        |     |        |        |              |        +---w input
        |     |        |        |              |        |  +---w subject
        |     |        |        |              |        |  |       bina\
   \ry
        |     |        |        |              |        |  +---w attrib\
   \utes?
        |     |        |        |              |        |          bina\
   \ry
        |     |        |        |              |        +--ro output
        |     |        |        |              |           +--ro certif\
   \icate-signing-request
        |     |        |        |              |                   bina\
   \ry
        |     |        |        |           +--:(keystore)
        |     |        |        |                    {keystore-suppo\
   \rted}?
        |     |        |        |              +--rw keystore-refere\
   \nce
        |     |        |        |                 +--rw asymmetric-k\
   \ey?
        |     |        |        |                 |       ks:asymmet\
   \ric-key-ref
        |     |        |        |                 +--rw certificate?\
   \      leafref
        |     |        |        +--rw server-authentication
        |     |        |        |  +--rw ssh-host-keys?
        |     |        |        |  |       ts:host-keys-ref
        |     |        |        |  |       {ts:ssh-host-keys}?
        |     |        |        |  +--rw ca-certs?
        |     |        |        |  |       ts:certificates-ref
        |     |        |        |  |       {sshcmn:ssh-x509-certs,ts:x5\
   \09-certificates}?
        |     |        |        |  +--rw server-certs?
        |     |        |        |          ts:certificates-ref
```

```
     |     |        |        |          {sshcmn:ssh-x509-certs,ts:x5\
  \09-certificates}?
     |     |        |        +--rw transport-params
     |     |        |        |       {ssh-client-transport-params-co\
  \nfig}?
     |     |        |        |  +--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 keepalives!
     |     |        |                {ssh-client-keepalives}?
     |     |        |           +--rw max-wait?       uint16
     |     |        |           +--rw max-attempts?   uint8
     |     |        +--:(tls) {tls-initiate}?
     |     |           +--rw tls
     |     |              +--rw tcp-client-parameters
     |     |              |  +--rw remote-address    inet:host
     |     |              |  +--rw remote-port?      inet:port-number
     |     |              |  +--rw local-address?    inet:ip-address
     |     |              |  |       {local-binding-supported}?
     |     |              |  +--rw local-port?       inet:port-number
     |     |              |  |       {local-binding-supported}?
     |     |              |  +--rw keepalives!
     |     |              |          {keepalives-supported}?
     |     |              |     +--rw idle-time         uint16
     |     |              |     +--rw max-probes        uint16
     |     |              |     +--rw probe-interval    uint16
     |     |              +--rw tls-client-parameters
     |     |                 +--rw client-identity
     |     |                 |  +--rw (local-or-keystore)
     |     |                 |     +--:(local)
     |     |                 |     |        {local-definitions-suppo\
  \rted}?
     |     |                 |     |  +--rw local-definition
     |     |                 |     |     +--rw algorithm
     |     |                 |     |     |       asymmetric-key-algo\
  \rithm-t
     |     |                 |     |     +--rw public-key
     |     |                 |     |     |       binary
     |     |                 |     |     +--rw (private-key-type)
     |     |                 |     |     |  +--:(private-key)
     |     |                 |     |     |  |  +--rw private-key?
```

```
        |     |                 |     |     |  |          binary
        |     |                 |     |     |  +--:(hidden-private-key)
        |     |                 |     |     |  |  +--rw hidden-private-\
   \key?
        |     |                 |     |     |  |          empty
        |     |                 |     |     |  +--:(encrypted-private-k\
   \ey)
        |     |                 |     |     |     +--rw encrypted-priva\
   \te-key
        |     |                 |     |     |        +--rw (key-type)
        |     |                 |     |     |        |  +--:(symmetric-\
   \key-ref)
        |     |                 |     |     |        |  |  +--rw symmet\
   \ric-key-ref?    leafref
        |     |                 |     |     |        |  |          {key\
   \store-supported}?
        |     |                 |     |     |        |  +--:(asymmetric\
   \-key-ref)
        |     |                 |     |     |        |     +--rw asymme\
   \tric-key-ref?   leafref
        |     |                 |     |     |        |             {key\
   \store-supported}?
        |     |                 |     |     |        +--rw value?
        |     |                 |     |     |                binary
        |     |                 |     |     +--rw cert?
        |     |                 |     |     |       end-entity-cert-cms
        |     |                 |     |     +---n certificate-expiration
        |     |                 |     |     |  +-- expiration-date
        |     |                 |     |     |          yang:date-and-ti\
   \me
        |     |                 |     |     +---x generate-certificate-\
   \signing-request
        |     |                 |     |        +---w input
        |     |                 |     |        |  +---w subject
        |     |                 |     |        |  |       binary
        |     |                 |     |        |  +---w attributes?
        |     |                 |     |        |          binary
        |     |                 |     |        +--ro output
        |     |                 |     |           +--ro certificate-sig\
   \ning-request
        |     |                 |     |                   binary
        |     |                 |     +--:(keystore)
        |     |                 |              {keystore-supported}?
        |     |                 |        +--rw keystore-reference
        |     |                 |           +--rw asymmetric-key?
        |     |                 |           |       ks:asymmetric-key-r\
   \ef
        |     |                 |           +--rw certificate?      lea\
```

```
   \fref
        |     |                 +--rw server-authentication
        |     |                 |  +--rw ca-certs?
        |     |                 |  |       ts:certificates-ref
        |     |                 |  |       {ts:x509-certificates}?
        |     |                 |  +--rw server-certs?
        |     |                 |          ts:certificates-ref
        |     |                 |          {ts:x509-certificates}?
        |     |                 +--rw hello-params
        |     |                 |       {tls-client-hello-params-config\
   \}?
        |     |                 |  +--rw tls-versions
        |     |                 |  |  +--rw tls-version*   identityref
        |     |                 |  +--rw cipher-suites
        |     |                 |     +--rw cipher-suite*   identityref
        |     |                 +--rw keepalives!
        |     |                       {tls-client-keepalives}?
        |     |                    +--rw max-wait?       uint16
        |     |                    +--rw max-attempts?   uint8
        |     +--rw connection-type
        |     |  +--rw (connection-type)
        |     |     +--:(persistent-connection)
        |     |     |  +--rw persistent!
        |     |     +--:(periodic-connection)
        |     |        +--rw periodic!
        |     |           +--rw period?         uint16
        |     |           +--rw anchor-time?    yang:date-and-time
        |     |           +--rw idle-timeout?   uint16
        |     +--rw reconnect-strategy
        |        +--rw start-with?     enumeration
        |        +--rw max-attempts?   uint8
        +--rw listen! {ssh-listen or tls-listen}?
           +--rw idle-timeout?   uint16
           +--rw endpoint* [name]
              +--rw name         string
              +--rw (transport)
                 +--:(ssh) {ssh-listen}?
                 |  +--rw ssh
                 |     +--rw tcp-server-parameters
                 |     |  +--rw local-address    inet:ip-address
                 |     |  +--rw local-port?      inet:port-number
                 |     |  +--rw keepalives! {keepalives-supported}?
                 |     |     +--rw idle-time         uint16
                 |     |     +--rw max-probes        uint16
                 |     |     +--rw probe-interval    uint16
                 |     +--rw ssh-client-parameters
                 |        +--rw client-identity
                 |        |  +--rw username?            string
```

```
                 |        |  +--rw (auth-type)
                 |        |     +--:(password)
                 |        |     |  +--rw password?      string
                 |        |     +--:(public-key)
                 |        |     |  +--rw public-key
                 |        |     |     +--rw (local-or-keystore)
                 |        |     |        +--:(local)
                 |        |     |        |        {local-definitions-su\
   \pported}?
                 |        |     |        |  +--rw local-definition
                 |        |     |        |     +--rw algorithm
                 |        |     |        |     |       asymmetric-key-a\
   \lgorithm-t
                 |        |     |        |     +--rw public-key
                 |        |     |        |     |       binary
                 |        |     |        |     +--rw (private-key-type)
                 |        |     |        |        +--:(private-key)
                 |        |     |        |        |  +--rw private-key?
                 |        |     |        |        |          binary
                 |        |     |        |        +--:(hidden-private-k\
   \ey)
                 |        |     |        |        |  +--rw hidden-priva\
   \te-key?
                 |        |     |        |        |          empty
                 |        |     |        |        +--:(encrypted-privat\
   \e-key)
                 |        |     |        |           +--rw encrypted-pr\
   \ivate-key
                 |        |     |        |              +--rw (key-type)
                 |        |     |        |              |  +--:(symmetr\
   \ic-key-ref)
                 |        |     |        |              |  |  +--rw sym\
   \metric-key-ref?    leafref
                 |        |     |        |              |  |          {\
   \keystore-supported}?
                 |        |     |        |              |  +--:(asymmet\
   \ric-key-ref)
                 |        |     |        |              |     +--rw asy\
   \mmetric-key-ref?   leafref
                 |        |     |        |              |             {\
   \keystore-supported}?
                 |        |     |        |              +--rw value?
                 |        |     |        |                      binary
                 |        |     |        +--:(keystore)
                 |        |     |                 {keystore-supported}?
                 |        |     |           +--rw keystore-reference?
                 |        |     |                   ks:asymmetric-key-r\
   \ef
```

```
                 |        |     +--:(certificate)
                 |        |        +--rw certificate
                 |        |                {sshcmn:ssh-x509-certs}?
                 |        |           +--rw (local-or-keystore)
                 |        |              +--:(local)
                 |        |              |        {local-definitions-su\
   \pported}?
                 |        |              |  +--rw local-definition
                 |        |              |     +--rw algorithm
                 |        |              |     |       asymmetric-key-a\
   \lgorithm-t
                 |        |              |     +--rw public-key
                 |        |              |     |       binary
                 |        |              |     +--rw (private-key-type)
                 |        |              |     |  +--:(private-key)
                 |        |              |     |  |  +--rw private-key?
                 |        |              |     |  |          binary
                 |        |              |     |  +--:(hidden-private-k\
   \ey)
                 |        |              |     |  |  +--rw hidden-priva\
   \te-key?
                 |        |              |     |  |          empty
                 |        |              |     |  +--:(encrypted-privat\
   \e-key)
                 |        |              |     |     +--rw encrypted-pr\
   \ivate-key
                 |        |              |     |        +--rw (key-type)
                 |        |              |     |        |  +--:(symmetr\
   \ic-key-ref)
                 |        |              |     |        |  |  +--rw sym\
   \metric-key-ref?    leafref
                 |        |              |     |        |  |          {\
   \keystore-supported}?
                 |        |              |     |        |  +--:(asymmet\
   \ric-key-ref)
                 |        |              |     |        |     +--rw asy\
   \mmetric-key-ref?   leafref
                 |        |              |     |        |             {\
   \keystore-supported}?
                 |        |              |     |        +--rw value?
                 |        |              |     |                binary
                 |        |              |     +--rw cert?
                 |        |              |     |       end-entity-cert-\
   \cms
                 |        |              |     +---n certificate-expira\
   \tion
                 |        |              |     |  +-- expiration-date
                 |        |              |     |          yang:date-and\
```

```
\-time
              |        |              |     +---x generate-certifica\
\te-signing-request
              |        |              |        +---w input
              |        |              |        |  +---w subject
              |        |              |        |  |       binary
              |        |              |        |  +---w attributes?
              |        |              |        |          binary
              |        |              |        +--ro output
              |        |              |           +--ro certificate-\
\signing-request
              |        |              |                   binary
              |        |             +--:(keystore)
              |        |                      {keystore-supported}?
              |        |                +--rw keystore-reference
              |        |                   +--rw asymmetric-key?
              |        |                   |       ks:asymmetric-ke\
\y-ref
              |        |                   +--rw certificate?      \
\leafref
              |        +--rw server-authentication
              |        |  +--rw ssh-host-keys?   ts:host-keys-ref
              |        |  |       {ts:ssh-host-keys}?
              |        |  +--rw ca-certs?        ts:certificates-ref
              |        |  |       {sshcmn:ssh-x509-certs,ts:x509-cer\
\tificates}?
              |        |  +--rw server-certs?    ts:certificates-ref
              |        |          {sshcmn:ssh-x509-certs,ts:x509-cer\
\tificates}?
              |        +--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 keepalives! {ssh-client-keepalives}?
              |           +--rw max-wait?       uint16
              |           +--rw max-attempts?   uint8
              +--:(tls) {tls-listen}?
                 +--rw tls
                    +--rw tcp-server-parameters
                    |  +--rw local-address    inet:ip-address
                    |  +--rw local-port?      inet:port-number
                    |  +--rw keepalives! {keepalives-supported}?
```

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

```
                           |     |           +--ro certificate-signing-r\
   \equest
                           |     |                   binary
                           |     +--:(keystore) {keystore-supported}?
                           |        +--rw keystore-reference
                           |           +--rw asymmetric-key?
                           |           |       ks:asymmetric-key-ref
                           |           +--rw certificate?      leafref
                           +--rw server-authentication
                           |  +--rw ca-certs?       ts:certificates-ref
                           |  |       {ts:x509-certificates}?
                           |  +--rw server-certs?   ts:certificates-ref
                           |          {ts:x509-certificates}?
                           +--rw hello-params
                           |       {tls-client-hello-params-config}?
                           |  +--rw tls-versions
                           |  |  +--rw tls-version*   identityref
                           |  +--rw cipher-suites
                           |     +--rw cipher-suite*   identityref
                           +--rw keepalives! {tls-client-keepalives}?
                              +--rw max-wait?       uint16
                              +--rw max-attempts?   uint8
```

A.2. Expanded Tree Diagram for 'ietf-netconf-server'

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

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

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

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

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

```
        |        |     |  +--rw keepalives! {keepalives-supported}?
        |        |     |     +--rw idle-time         uint16
        |        |     |     +--rw max-probes        uint16
        |        |     |     +--rw probe-interval    uint16
        |        |     +--rw ssh-server-parameters
        |        |        +--rw server-identity
        |        |        |  +--rw host-key* [name]
        |        |        |     +--rw name                       string
        |        |        |     +--rw (host-key-type)
        |        |        |        +--:(public-key)
        |        |        |        |  +--rw public-key
        |        |        |        |     +--rw (local-or-keystore)
        |        |        |        |        +--:(local)
        |        |        |        |        |        {local-definitions\
   \-supported}?
        |        |        |        |        |  +--rw local-definition
        |        |        |        |        |     +--rw algorithm
        |        |        |        |        |     |       asymmetric-ke\
   \y-algorithm-t
        |        |        |        |        |     +--rw public-key
        |        |        |        |        |     |       binary
        |        |        |        |        |     +--rw (private-key-ty\
   \pe)
        |        |        |        |        |        +--:(private-key)
        |        |        |        |        |        |  +--rw private-k\
   \ey?
        |        |        |        |        |        |          binary
        |        |        |        |        |        +--:(hidden-privat\
   \e-key)
        |        |        |        |        |        |  +--rw hidden-pr\
   \ivate-key?
        |        |        |        |        |        |          empty
        |        |        |        |        |        +--:(encrypted-pri\
   \vate-key)
        |        |        |        |        |           +--rw encrypted\
   \-private-key
        |        |        |        |        |              +--rw (key-t\
   \ype)
        |        |        |        |        |              |  +--:(symm\
   \etric-key-ref)
        |        |        |        |        |              |  |  +--rw \
   \symmetric-key-ref?    leafref
        |        |        |        |        |              |  |        \
   \  {keystore-supported}?
        |        |        |        |        |              |  +--:(asym\
   \metric-key-ref)
        |        |        |        |        |              |     +--rw \
   \asymmetric-key-ref?   leafref
```

```
        |        |        |        |        |              |           \
   \  {keystore-supported}?
        |        |        |        |        |              +--rw value?
        |        |        |        |        |                      bina\
   \ry
        |        |        |        |        +--:(keystore)
        |        |        |        |              {keystore-supporte\
   \d}?
        |        |        |        |           +--rw keystore-reference?
        |        |        |        |                   ks:asymmetric-ke\
   \y-ref
        |        |        |        +--:(certificate)
        |        |        |           +--rw certificate
        |        |        |                   {sshcmn:ssh-x509-certs}?
        |        |        |              +--rw (local-or-keystore)
        |        |        |                 +--:(local)
        |        |        |                 |        {local-definitions\
   \-supported}?
        |        |        |                 |  +--rw local-definition
        |        |        |                 |     +--rw algorithm
        |        |        |                 |     |       asymmetric-ke\
   \y-algorithm-t
        |        |        |                 |     +--rw public-key
        |        |        |                 |     |       binary
        |        |        |                 |     +--rw (private-key-ty\
   \pe)
        |        |        |                 |     |  +--:(private-key)
        |        |        |                 |     |  |  +--rw private-k\
   \ey?
        |        |        |                 |     |  |          binary
        |        |        |                 |     |  +--:(hidden-privat\
   \e-key)
        |        |        |                 |     |  |  +--rw hidden-pr\
   \ivate-key?
        |        |        |                 |     |  |          empty
        |        |        |                 |     |  +--:(encrypted-pri\
   \vate-key)
        |        |        |                 |     |     +--rw encrypted\
   \-private-key
        |        |        |                 |     |        +--rw (key-t\
   \ype)
        |        |        |                 |     |        |  +--:(symm\
   \etric-key-ref)
        |        |        |                 |     |        |  |  +--rw \
   \symmetric-key-ref?    leafref
        |        |        |                 |     |        |  |        \
   \  {keystore-supported}?
        |        |        |                 |     |        |  +--:(asym\
```

```
   \metric-key-ref)
        |        |        |                 |     |        |     +--rw \
   \asymmetric-key-ref?   leafref
        |        |        |                 |     |        |           \
   \  {keystore-supported}?
        |        |        |                 |     |           +--rw value?
        |        |        |                 |     |                   bina\
   \ry
        |        |        |                 |     +--rw cert?
        |        |        |                 |     |          end-entity-ce\
   \rt-cms
        |        |        |                 |     +---n certificate-exp\
   \iration
        |        |        |                 |     |  +-- expiration-date
        |        |        |                 |     |             yang:date-\
   \and-time
        |        |        |                 |     +---x generate-certif\
   \icate-signing-request
        |        |        |                 |        +---w input
        |        |        |                 |        |  +---w subject
        |        |        |                 |        |  |       binary
        |        |        |                 |        |  +---w attribute\
   \s?
        |        |        |                 |        |          binary
        |        |        |                 |        +--ro output
        |        |        |                 |           +--ro certifica\
   \te-signing-request
        |        |        |                 |                   binary
        |        |        |                 +--:(keystore)
        |        |        |                          {keystore-supporte\
   \d}?
        |        |        |                    +--rw keystore-reference
        |        |        |                       +--rw asymmetric-key?
        |        |        |                       |       ks:asymmetric\
   \-key-ref
        |        |        |                       +--rw certificate?   \
   \   leafref
        |        |       +--rw client-authentication
        |        |       |  +--rw supported-authentication-methods
        |        |       |  |  +--rw publickey?   empty
        |        |       |  |  +--rw passsword?   empty
        |        |       |  |  +--rw hostbased?   empty
        |        |       |  |  +--rw none?        empty
        |        |       |  |  +--rw other*       string
        |        |       |  +--rw (local-or-external)
        |        |       |     +--:(local)
        |        |       |     |         {local-client-auth-supported}?
        |        |       |     |  +--rw users
```

```
        |        |        |     |     +--rw user* [name]
        |        |        |     |        +--rw name              string
        |        |        |     |        +--rw password?
        |        |        |     |        |       ianach:crypt-hash
        |        |        |     |        +--rw authorized-key* [name]
        |        |        |     |           +--rw name         string
        |        |        |     |           +--rw algorithm    string
        |        |        |     |           +--rw key-data     binary
        |        |        |     +--:(external)
        |        |        |              {external-client-auth-supporte\
   \d}?
        |        |        |        +--rw client-auth-defined-elsewhere?
        |        |        |                empty
        |        |        +--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 keepalives! {ssh-server-keepalives}?
        |        |           +--rw max-wait?       uint16
        |        |           +--rw max-attempts?   uint8
        |        +--:(tls) {tls-listen}?
        |           +--rw tls
        |              +--rw tcp-server-parameters
        |              |  +--rw local-address    inet:ip-address
        |              |  +--rw local-port?      inet:port-number
        |              |  +--rw keepalives! {keepalives-supported}?
        |              |     +--rw idle-time         uint16
        |              |     +--rw max-probes        uint16
        |              |     +--rw probe-interval    uint16
        |              +--rw tls-server-parameters
        |                 +--rw server-identity
        |                 |  +--rw (local-or-keystore)
        |                 |     +--:(local)
        |                 |     |        {local-definitions-supported}?
        |                 |     |  +--rw local-definition
        |                 |     |     +--rw algorithm
        |                 |     |     |       asymmetric-key-algorithm-t
        |                 |     |     +--rw public-key
        |                 |     |     |       binary
        |                 |     |     +--rw (private-key-type)
        |                 |     |     |  +--:(private-key)
        |                 |     |     |  |  +--rw private-key?
```

```
        |                  |     |     |  |         binary
        |                  |     |     |  +--:(hidden-private-key)
        |                  |     |     |  |  +--rw hidden-private-key?
        |                  |     |     |  |          empty
        |                  |     |     |  +--:(encrypted-private-key)
        |                  |     |     |     +--rw encrypted-private-key
        |                  |     |     |        +--rw (key-type)
        |                  |     |     |        |  +--:(symmetric-key-re\
   \f)
        |                  |     |     |        |  |  +--rw symmetric-ke\
   \y-ref?    leafref
        |                  |     |     |        |  |          {keystore-\
   \supported}?
        |                  |     |     |        |  +--:(asymmetric-key-r\
   \ef)
        |                  |     |     |        |     +--rw asymmetric-k\
   \ey-ref?   leafref
        |                  |     |     |        |             {keystore-\
   \supported}?
        |                  |     |     |        +--rw value?
        |                  |     |     |                 binary
        |                  |     |     +--rw cert?
        |                  |     |     |       end-entity-cert-cms
        |                  |     |     +---n certificate-expiration
        |                  |     |     |  +-- expiration-date
        |                  |     |     |          yang:date-and-time
        |                  |     |     +---x generate-certificate-signin\
   \g-request
        |                  |     |        +---w input
        |                  |     |        |  +---w subject       binary
        |                  |     |        |  +---w attributes?   binary
        |                  |     |        +--ro output
        |                  |     |           +--ro certificate-signing-r\
   \equest
        |                  |     |                   binary
        |                  |     +--:(keystore) {keystore-supported}?
        |                  |        +--rw keystore-reference
        |                  |           +--rw asymmetric-key?
        |                  |           |       ks:asymmetric-key-ref
        |                  |           +--rw certificate?      leafref
        |                  +--rw client-authentication!
        |                  |  +--rw (required-or-optional)
        |                  |  |  +--:(required)
        |                  |  |  |  +--rw required?
        |                  |  |  |          empty
        |                  |  |  +--:(optional)
        |                  |  |     +--rw optional?
        |                  |  |             empty
```

```
        |                 |  +--rw (local-or-external)
        |                 |  |  +--:(local)
        |                 |  |  |        {local-client-auth-supported}?
        |                 |  |  |  +--rw ca-certs?
        |                 |  |  |  |       ts:certificates-ref
        |                 |  |  |  |       {ts:x509-certificates}?
        |                 |  |  |  +--rw client-certs?
        |                 |  |  |          ts:certificates-ref
        |                 |  |  |          {ts:x509-certificates}?
        |                 |  |  +--:(external)
        |                 |  |           {external-client-auth-supporte\
   \d}?
        |                 |  |     +--rw client-auth-defined-elsewhere?
        |                 |  |             empty
        |                 |  +--rw cert-maps
        |                 |     +--rw cert-to-name* [id]
        |                 |        +--rw id             uint32
        |                 |        +--rw fingerprint
        |                 |        |       x509c2n:tls-fingerprint
        |                 |        +--rw map-type       identityref
        |                 |        +--rw name           string
        |                 +--rw hello-params
        |                 |       {tls-server-hello-params-config}?
        |                 |  +--rw tls-versions
        |                 |  |  +--rw tls-version*   identityref
        |                 |  +--rw cipher-suites
        |                 |     +--rw cipher-suite*   identityref
        |                 +--rw keepalives! {tls-server-keepalives}?
        |                    +--rw max-wait?       uint16
        |                    +--rw max-attempts?   uint8
        +--rw call-home! {ssh-call-home or tls-call-home}?
           +--rw netconf-client* [name]
              +--rw name                  string
              +--rw endpoints
              |  +--rw endpoint* [name]
              |     +--rw name         string
              |     +--rw (transport)
              |        +--:(ssh) {ssh-call-home}?
              |        |  +--rw ssh
              |        |     +--rw tcp-client-parameters
              |        |     |  +--rw remote-address    inet:host
              |        |     |  +--rw remote-port?      inet:port-number
              |        |     |  +--rw local-address?    inet:ip-address
              |        |     |  |       {local-binding-supported}?
              |        |     |  +--rw local-port?       inet:port-number
              |        |     |  |       {local-binding-supported}?
              |        |     |  +--rw keepalives!
              |        |     |          {keepalives-supported}?
```

```
             |        |      |     +--rw idle-time         uint16
             |        |      |     +--rw max-probes        uint16
             |        |      |     +--rw probe-interval    uint16
             |        |     +--rw ssh-server-parameters
             |        |        +--rw server-identity
             |        |        |  +--rw host-key* [name]
             |        |        |     +--rw name                  string
             |        |        |     +--rw (host-key-type)
             |        |        |        +--:(public-key)
             |        |        |        |  +--rw public-key
             |        |        |        |     +--rw (local-or-keystore)
             |        |        |        |        +--:(local)
             |        |        |        |        |        {local-defin\
   \itions-supported}?
             |        |        |        |        |  +--rw local-defini\
   \tion
             |        |        |        |        |     +--rw algorithm
             |        |        |        |        |     |       asymmet\
   \ric-key-algorithm-t
             |        |        |        |        |     +--rw public-key
             |        |        |        |        |     |       binary
             |        |        |        |        |     +--rw (private-\
   \key-type)
             |        |        |        |        |        +--:(private\
   \-key)
             |        |        |        |        |        |  +--rw pri\
   \vate-key?
             |        |        |        |        |        |          b\
   \inary
             |        |        |        |        |        +--:(hidden-\
   \private-key)
             |        |        |        |        |        |  +--rw hid\
   \den-private-key?
             |        |        |        |        |        |          e\
   \mpty
             |        |        |        |        |        +--:(encrypt\
   \ed-private-key)
             |        |        |        |        |           +--rw enc\
   \rypted-private-key
             |        |        |        |        |              +--rw \
   \(key-type)
             |        |        |        |        |              |  +--\
   \:(symmetric-key-ref)
             |        |        |        |        |              |  |  \
   \+--rw symmetric-key-ref?    leafref
             |        |        |        |        |              |  |  \
   \        {keystore-supported}?
             |        |        |        |        |              |  +--\
```

```
   \:(asymmetric-key-ref)
              |        |        |        |        |              |     \
   \+--rw asymmetric-key-ref?   leafref
              |        |        |        |        |              |     \
   \        {keystore-supported}?
              |        |        |        |        |              +--rw \
   \value?
              |        |        |        |        |                    \
   \  binary
              |        |        |        |        +--:(keystore)
              |        |        |        |                 {keystore-su\
   \pported}?
              |        |        |        |           +--rw keystore-ref\
   \erence?
              |        |        |        |                   ks:asymmet\
   \ric-key-ref
              |        |        |        +--:(certificate)
              |        |        |           +--rw certificate
              |        |        |                   {sshcmn:ssh-x509-ce\
   \rts}?
              |        |        |              +--rw (local-or-keystore)
              |        |        |                 +--:(local)
              |        |        |                 |        {local-defin\
   \itions-supported}?
              |        |        |                 |  +--rw local-defini\
   \tion
              |        |        |                 |     +--rw algorithm
              |        |        |                 |     |       asymmet\
   \ric-key-algorithm-t
              |        |        |                 |     +--rw public-key
              |        |        |                 |     |       binary
              |        |        |                 |     +--rw (private-\
   \key-type)
              |        |        |                 |     |  +--:(private\
   \-key)
              |        |        |                 |     |  |  +--rw pri\
   \vate-key?
              |        |        |                 |     |  |          b\
   \inary
              |        |        |                 |     |  +--:(hidden-\
   \private-key)
              |        |        |                 |     |  |  +--rw hid\
   \den-private-key?
              |        |        |                 |     |  |          e\
   \mpty
              |        |        |                 |     |  +--:(encrypt\
   \ed-private-key)
              |        |        |                 |     |     +--rw enc\
```

```
   \rypted-private-key
              |        |        |                 |     |        +--rw \
   \(key-type)
              |        |        |                 |     |        |  +--\
   \:(symmetric-key-ref)
              |        |        |                 |     |        |  |  \
   \+--rw symmetric-key-ref?    leafref
              |        |        |                 |     |        |  |  \
   \        {keystore-supported}?
              |        |        |                 |     |        |  +--\
   \:(asymmetric-key-ref)
              |        |        |                 |     |        |     \
   \+--rw asymmetric-key-ref?   leafref
              |        |        |                 |     |        |     \
   \        {keystore-supported}?
              |        |        |                 |     |        +--rw \
   \value?
              |        |        |                 |     |              \
   \  binary
              |        |        |                 |     +--rw cert?
              |        |        |                 |     |       end-ent\
   \ity-cert-cms
              |        |        |                 |     +---n certifica\
   \te-expiration
              |        |        |                 |     |  +-- expirati\
   \on-date
              |        |        |                 |     |           yang\
   \:date-and-time
              |        |        |                 |     +---x generate-\
   \certificate-signing-request
              |        |        |                 |        +---w input
              |        |        |                 |        |  +---w sub\
   \ject
              |        |        |                 |        |  |       b\
   \inary
              |        |        |                 |        |  +---w att\
   \ributes?
              |        |        |                 |        |          b\
   \inary
              |        |        |                 |        +--ro output
              |        |        |                 |           +--ro cer\
   \tificate-signing-request
              |        |        |                 |                   b\
   \inary
              |        |        |                 +--:(keystore)
              |        |        |                          {keystore-su\
   \pported}?
              |        |        |                    +--rw keystore-ref\
```

```
   \erence
              |        |        |                       +--rw asymmetri\
   \c-key?
              |        |        |                       |       ks:asym\
   \metric-key-ref
              |        |        |                       +--rw certifica\
   \te?      leafref
              |        |        +--rw client-authentication
              |        |        |  +--rw supported-authentication-metho\
   \ds
              |        |        |  |  +--rw publickey?   empty
              |        |        |  |  +--rw passsword?   empty
              |        |        |  |  +--rw hostbased?   empty
              |        |        |  |  +--rw none?        empty
              |        |        |  |  +--rw other*       string
              |        |        |  +--rw (local-or-external)
              |        |        |     +--:(local)
              |        |        |     |        {local-client-auth-suppo\
   \rted}?
              |        |        |     |  +--rw users
              |        |        |     |     +--rw user* [name]
              |        |        |     |        +--rw name
              |        |        |     |        |       string
              |        |        |     |        +--rw password?
              |        |        |     |        |       ianach:crypt-hash
              |        |        |     |        +--rw authorized-key*
              |        |        |     |                [name]
              |        |        |     |           +--rw name
              |        |        |     |           |       string
              |        |        |     |           +--rw algorithm
              |        |        |     |           |       string
              |        |        |     |           +--rw key-data
              |        |        |     |                   binary
              |        |        |     +--:(external)
              |        |        |              {external-client-auth-su\
   \pported}?
              |        |        |        +--rw client-auth-defined-else\
   \where?
              |        |        |                empty
              |        |        +--rw transport-params
              |        |        |       {ssh-server-transport-params-co\
   \nfig}?
              |        |        |  +--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 keepalives!
              |        |                {ssh-server-keepalives}?
              |        |           +--rw max-wait?       uint16
              |        |           +--rw max-attempts?   uint8
              |        +--:(tls) {tls-call-home}?
              |           +--rw tls
              |              +--rw tcp-client-parameters
              |              |  +--rw remote-address    inet:host
              |              |  +--rw remote-port?      inet:port-number
              |              |  +--rw local-address?    inet:ip-address
              |              |  |       {local-binding-supported}?
              |              |  +--rw local-port?       inet:port-number
              |              |  |       {local-binding-supported}?
              |              |  +--rw keepalives!
              |              |          {keepalives-supported}?
              |              |     +--rw idle-time         uint16
              |              |     +--rw max-probes        uint16
              |              |     +--rw probe-interval    uint16
              |              +--rw tls-server-parameters
              |                 +--rw server-identity
              |                 |  +--rw (local-or-keystore)
              |                 |     +--:(local)
              |                 |     |        {local-definitions-suppo\
   \rted}?
              |                 |     |  +--rw local-definition
              |                 |     |     +--rw algorithm
              |                 |     |     |       asymmetric-key-algo\
   \rithm-t
              |                 |     |     +--rw public-key
              |                 |     |     |       binary
              |                 |     |     +--rw (private-key-type)
              |                 |     |     |  +--:(private-key)
              |                 |     |     |  |  +--rw private-key?
              |                 |     |     |  |          binary
              |                 |     |     |  +--:(hidden-private-key)
              |                 |     |     |  |  +--rw hidden-private-\
   \key?
              |                 |     |     |  |          empty
              |                 |     |     |  +--:(encrypted-private-k\
   \ey)
              |                 |     |     |     +--rw encrypted-priva\
   \te-key
              |                 |     |     |        +--rw (key-type)
              |                 |     |     |        |  +--:(symmetric-\
```

```
   \key-ref)
              |                 |     |     |        |  |  +--rw symmet\
   \ric-key-ref?    leafref
              |                 |     |     |        |  |          {key\
   \store-supported}?
              |                 |     |     |        |  +--:(asymmetric\
   \-key-ref)
              |                 |     |     |        |     +--rw asymme\
   \tric-key-ref?   leafref
              |                 |     |     |        |             {key\
   \store-supported}?
              |                 |     |     |        +--rw value?
              |                 |     |     |              binary
              |                 |     |     +--rw cert?
              |                 |     |     |       end-entity-cert-cms
              |                 |     |     +---n certificate-expiration
              |                 |     |     |  +-- expiration-date
              |                 |     |     |          yang:date-and-ti\
   \me
              |                 |     |     +---x generate-certificate-\
   \signing-request
              |                 |     |        +---w input
              |                 |     |        |  +---w subject
              |                 |     |        |  |       binary
              |                 |     |        |  +---w attributes?
              |                 |     |        |          binary
              |                 |     |        +--ro output
              |                 |     |           +--ro certificate-sig\
   \ning-request
              |                 |     |                   binary
              |                 |     +--:(keystore)
              |                 |              {keystore-supported}?
              |                 |        +--rw keystore-reference
              |                 |           +--rw asymmetric-key?
              |                 |           |       ks:asymmetric-key-r\
   \ef
              |                 |           +--rw certificate?      lea\
   \fref
              |                 +--rw client-authentication!
              |                 |  +--rw (required-or-optional)
              |                 |  |  +--:(required)
              |                 |  |  |  +--rw required?
              |                 |  |  |          empty
              |                 |  |  +--:(optional)
              |                 |  |     +--rw optional?
              |                 |  |             empty
              |                 |  +--rw (local-or-external)
              |                 |  |  +--:(local)
```

```
              |                 |  |  |        {local-client-auth-suppo\
   \rted}?
              |                 |  |  |  +--rw ca-certs?
              |                 |  |  |  |       ts:certificates-ref
              |                 |  |  |  |       {ts:x509-certificates}?
              |                 |  |  |  +--rw client-certs?
              |                 |  |  |          ts:certificates-ref
              |                 |  |  |          {ts:x509-certificates}?
              |                 |  |  +--:(external)
              |                 |  |           {external-client-auth-su\
   \pported}?
              |                 |  |     +--rw client-auth-defined-else\
   \where?
              |                 |  |             empty
              |                 |  +--rw cert-maps
              |                 |     +--rw cert-to-name* [id]
              |                 |        +--rw id             uint32
              |                 |        +--rw fingerprint
              |                 |        |       x509c2n:tls-fingerprint
              |                 |        +--rw map-type
              |                 |        |       identityref
              |                 |        +--rw name           string
              |                 +--rw hello-params
              |                 |       {tls-server-hello-params-config\
   \}?
              |                 |  +--rw tls-versions
              |                 |  |  +--rw tls-version*   identityref
              |                 |  +--rw cipher-suites
              |                 |     +--rw cipher-suite*   identityref
              |                 +--rw keepalives!
              |                         {tls-server-keepalives}?
              |                    +--rw max-wait?       uint16
              |                    +--rw max-attempts?   uint8
              +--rw connection-type
              |  +--rw (connection-type)
              |     +--:(persistent-connection)
              |     |  +--rw persistent!
              |     +--:(periodic-connection)
              |        +--rw periodic!
              |           +--rw period?         uint16
              |           +--rw anchor-time?    yang:date-and-time
              |           +--rw idle-timeout?   uint16
              +--rw reconnect-strategy
                 +--rw start-with?     enumeration
                 +--rw max-attempts?   uint8
```

Appendix B. Change Log

B.1. 00 to 01

- Renamed "keychain" to "keystore".

B.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.

B.3. 02 to 03

- Refined use of tls-client-grouping to add a must statement indicating that the TLS client must specify a client-certificate.

- Changed 'netconf-client' to be a grouping (not a container).

B.4. 03 to 04

- Added RFC 8174 to Requirements Language Section.

- Replaced refine statement in ietf-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.

B.5. 04 to 05

- Now tree diagrams reference ietf-netmod-yang-tree-diagrams

- Updated examples to inline key and certificates (no longer a leafref to keystore)

B.6. 05 to 06

- Fixed change log missing section issue.

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

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

B.7. 06 to 07

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

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

- Replaced "connection-type" choice default (persistent) with "mandatory true".

- Reduced the periodic-connection's "idle-timeout" from 5 to 2 minutes.

- Replaced reconnect-timeout with period/anchor-time combo.

B.8. 07 to 08

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

B.9. 08 to 09

- Corrected use of "mandatory true" for "address" leafs.

- Updated examples to reflect update to groupings defined in the keystore draft.

- Updated to use groupings defined in new TCP and HTTP drafts.

- Updated copyright date, boilerplate template, affiliation, and folding algorithm.

B.10. 09 to 10

- Reformatted YANG modules.

B.11. 10 to 11

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

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

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

- Moved "expanded" tree diagrams to the Appendix.

B.12. 11 to 12

- Removed the "Design Considerations" section.

- Removed the 'must' statement limiting keepalives in periodic connections.

- Updated models and examples to reflect removal of the "demux" containers in the imported models.

- Updated the "periodic-connnection" description statements to be more like the RESTCONF draft, especially where it described dropping the underlying TCP connection.

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

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

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

B.13. 12 to 13

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

B.14. 13 to 14

- Adjusting from change in TLS client model (removing the top-level 'certificate' container), by swapping refining-in a 'mandatory true' statement with a 'must' statement outside the 'uses' statement.

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

Acknowledgements

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

Author's Address

Kent Watsen
Watsen Networks

EMail: kent+ietf@watsen.net

NETCONF E. Voit
Internet-Draft Cisco Systems
Intended status: Standards Track A. Clemm
Expires: November 20, 2019 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

Copyright Notice

Table of Contents

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:

- An "error-type" node of "application".
- 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

- an "error-severity" of "error" (this MAY be included).
- 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

- 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/>.

[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/>.

[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.

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

Figure 2: Multiple subscriptions over a NETCONF session

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

```
<rpc message-id="102" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <establish-subscription
      xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications">
    <stream-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.

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

Figure 6: Interaction model for successful subscription modification

If the subscription being modified in Figure 6 is a datastore subscription as per [I-D.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

- 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

- Per Kent's request, added name attribute to artwork. This would be needed for an automated extraction.

B.8. v13 to v14

- Title change.

B.9. v11 to v13

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

B.10. v10 to v11

- Configured removed.

B.11. v09 to v10

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

B.12. v08 to v09

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

B.13. v07 to v08

- 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

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

Network Working Group
Internet-Draft
Updates: 6241, 7950 (if approved)
Intended status: Standards Track
Expires: April 20, 2019

M. Bjorklund
Tail-f Systems
J. Schoenwaelder
Jacobs University
P. Shafer
K. Watsen
Juniper Networks
R. Wilton
Cisco Systems

October 17, 2018

NETCONF Extensions to Support the Network Management Datastore Architecture
draft-ietf-netconf-nmda-netconf-08

Abstract

This document extends the NETCONF protocol defined in RFC 6241 in order to support the Network Management Datastore Architecture defined in RFC 8342.

This document updates both RFC 6241 and RFC 7950. The update to RFC 6241 adds new operations <get-data> and <edit-data>, and augments existing operations <lock>, <unlock>, and <validate>. The update to RFC 7950 requires the usage of I-D.ietf-netconf-rfc7895bis by NETCONF servers implementing the Network Management Datastore Architecture.

RFC Ed.: Please replace "I-D.ietf-netconf-rfc7895bis" above with its final RFC assignment and remove this note.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document extends the NETCONF protocol defined in [RFC6241] in order to support the Network Management Datastore Architecture (NMDA) defined in [RFC8342].

This document updates [RFC6241] in order to enable NETCONF clients to interact with all the datastores supported by a server implementing the NMDA. The update both adds new operations <get-data> and <edit-data>, and augments existing operations <lock>, <unlock>, and <validate>.

This document also updates [RFC7950] in order to enable NETCONF clients to both discover which datastores are supported by the

NETCONF server, as well as determine which modules are supported in each datastore. The update requires NETCONF servers implementing the NMDA to support [I-D.ietf-netconf-rfc7895bis].

1.1. Terminology

This document uses the terminology defined by the NMDA [RFC8342].

The following term is defined in [I-D.ietf-netconf-rfc7895bis]:

o YANG library content identifier

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

1.2. Tree Diagrams

Tree diagrams used in this document follow the notation defined in [RFC8340].

2. Datastore and YANG Library Requirements

RFC Ed.: Update 201X-XX-XX below with correct date.

An NMDA-compliant NETCONF server MUST implement the module "ietf-netconf-nmda" defined in this document, MUST support the operational state datastore, and it MUST implement at least revision 201X-XX-XX of the "ietf-yang-library" module defined in [I-D.ietf-netconf-rfc7895bis].

A NETCONF client can discover which datastores and YANG modules the server supports by reading the YANG library information from the operational state datastore.

The server MUST advertise the following capability in the <hello> message (line breaks and whitespaces are used for formatting reasons only):

```
  urn:ietf:params:netconf:capability:yang-library:1.1?
    revision=<date>&content-id=<content-id-value>
```

The parameter "revision" has the same value as the revision date of the "ietf-yang-library" module implemented by the server. This parameter MUST be present.

The parameter "content-id" contains the YANG library content identifier [I-D.ietf-netconf-rfc7895bis]. This parameter MUST be present.

With this mechanism, a client can cache the supported datastores and YANG modules for a server and only update the cache if the "content-id" value in the <hello> message changes.

This document updates [RFC7950], Section 5.6.4, to allow servers to advertise the capability :yang-library:1.1 instead of :yang-library:1.0, and to implement the subtree "/yang-library" [I-D.ietf-netconf-rfc7895bis] instead of "/modules-state".

3. NETCONF Extensions

This section describes the NETCONF extensions needed to support the NMDA. These changes are defined in a new YANG ([RFC7950]) module "ietf-netconf-nmda".

These changes include the use of source and target parameters based on the "datastore" identity defined in the "ietf-datastores" module [RFC8342]. The use of identities allows future expansion in a way that the choice-based strategy from the original operations (e.g., <get-config>, <edit-config>) does not.

3.1. New NETCONF Operations

Two new operations <get-data> and <edit-data> are defined in this document in order to support the NMDA. These operations are similar to the <get-config> and <edit-config> operations but they can work on an extensible set of datastores.

3.1.1. The <get-data> Operation

The <get-data> operation retrieves data from a specific NMDA datastore. This operation is similar to NETCONF's <get-config> operation defined in [RFC6241], but it adds the flexibility to select the source datastore.

```
   +---x get-data
      +---w input
      |  +---w datastore                        ds:datastore-ref
      |  +---w (filter-spec)?
      |  |  +--:(subtree-filter)
      |  |  |  +---w subtree-filter?          <anydata>
      |  |  +--:(xpath-filter)
      |  |     +---w xpath-filter?            yang:xpath1.0 {nc:xpath}?
      |  +---w config-filter?                 boolean
      |  +---w (origin-filters)? {origin}?
      |  |  +--:(origin-filter)
      |  |  |  +---w origin-filter*           or:origin-ref
      |  |  +--:(negated-origin-filter)
      |  |     +---w negated-origin-filter*   or:origin-ref
      |  +---w max-depth?                     union
      |  +---w with-origin?                   empty {origin}?
      |  +---w with-defaults?                 with-defaults-mode
      +--ro output
         +--ro data?   <anydata>
```

The "datastore" parameter indicates the datastore which is the source of the data to be retrieved. This is a datastore identity.

The <get-data> operation accepts a content filter parameter, similar to the "filter" parameter of <get-config>, but using explicit nodes for subtree filtering ("subtree-filter") and XPath filtering ("xpath-filter").

The "config-filter" parameter can be used to retrieve only "config true" or "config false" nodes.

The "origin-filter" parameter, which can be present multiple times, selects nodes equal to or derived from any of the given values. The "negated-origin-filter", which can be present multiple times, selects nodes that do are not equal or derived from any of the given values. The "origin-filter" and "negated-origin-filter" parameters cannot be used together.

The "max-depth" parameter can be used by the client to limit the number of sub-tree levels that are returned in the reply.

3.1.1.1. Origin Metadata Attribute

The <get-data> operation defines a parameter named "with-origin", which if present, requests that the server includes "origin" metadata annotations in its response, as detailed in the NMDA. This parameter is only valid for the operational state datastore and any datastores with identities derived from the "operational" identity. Otherwise, if an invalid datastore is specified then an error is returned, as specified in "ietf-netconf-nmda" (see Section 4). Note that "origin" metadata annotations are not included in a response unless a client explicitly requests them.

Data in the operational state datastore can come from multiple sources. The server should return the most accurate value for the "origin" metadata annotation as possible, indicating the source of the operational value, as specified in Section 5.3.4 of [RFC8342].

When encoding the origin metadata annotation for a hierarchy of returned nodes, the annotation may be omitted for a child node when the value matches that of the parent node, as described in the "ietf-origin" YANG module [RFC8342].

The "with-origin" parameter is OPTIONAL to support. It is identified with the feature "origin".

3.1.1.2. With-defaults interactions

If the "with-defaults" capability is supported by the server, then the "with-defaults" parameter, defined in [RFC6243], is supported for <get-data> operations that target conventional configuration datastores.

The "with-defaults" parameter is OPTIONAL to support for <get-data> operations that target <operational>. The associated capability to indicate a server’s support is identified with the URI:

    urn:ietf:params:netconf:capability:with-operational-defaults:1.0

If the "with-defaults" parameter is supported for <get-data> operations on <operational>, then all retrieval modes specified in either the ’basic-mode’ or ’also-supported’ parameters of the "with-defaults" capability are permitted. The behavior of the "with-defaults" parameter for <operational> is defined as below:

- o If no "with-defaults" parameter is specified, or if it is set to "explicit", "report-all", or "report-all-tagged", then the "in use" values, as defined in [RFC8342] section 5.3, are returned from the operational state datastore, even if a node happens to have a default statement in the YANG module, and this default value is being used by the server. If the "with-defaults" parameter is set to "report-all-tagged", any values that match the schema default are tagged with additional metadata, as described in [RFC6243] section 3.4.

o If the "with-defaults" parameter is set to "trim", all "in use" values are returned, except that the output is filtered to exclude any values that match the default defined in the YANG schema.

Support for "with-defaults" in <get-data> operations on any datastore not defined in [RFC8342] should be defined by the specification for the datastore.

3.1.1.3. Example: Retrieving an entire subtree from <running>

The following example shows the <get-data> version of the <get-config> example shown in Section 7.1 of [RFC6241], which selects the entire "/users" subtree:

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-nmda"
            xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
    <datastore>ds:running</datastore>
    <subtree-filter>
      <top xmlns="http://example.com/schema/1.2/config">
        <users/>
      </top>
    </subtree-filter>
  </get-data>
</rpc>

<rpc-reply message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-nmda">
    <top xmlns="http://example.com/schema/1.2/config">
      <users>
        <user>
          <name>root</name>
          <type>superuser</type>
          <full-name>Charlie Root</full-name>
          <company-info>
            <dept>1</dept>
            <id>1</id>
          </company-info>
        </user>
        <!-- additional <user> elements appear here... -->
      </users>
    </top>
  </data>
</rpc-reply>
```

3.1.1.4. Example: Retrieving a filtered subtree from <operational>

The following example shows how the "origin-filter" can be used to retrieve nodes from <operational>. The example uses the fictional data model defined in Appendix C of [RFC8342].

```
<rpc message-id="102"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-nmda"
            xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores"
            xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin">
    <datastore>ds:operational</datastore>
    <subtree-filter>
      <bgp xmlns="http://example.com/ns/bgp"/>
    </subtree-filter>
    <origin-filter>or:intended</origin-filter>
    <origin-filter>or:system</origin-filter>
    <with-origin/>
  </get-data>
</rpc>

<rpc-reply message-id="102"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-nmda">
    <bgp xmlns="http://example.com/ns/bgp"
         xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
         or:origin="or:intended">
      <peer>
        <name>2001:db8::2:3</name>
        <local-port or:origin="or:system">60794</local-port>
        <state>established</state>
      </peer>
    </bgp>
  </data>
</rpc-reply>
```

3.1.2. The <edit-data> Operation

The <edit-data> operation changes the contents of a writable datastore, similar to the <edit-config> operation defined in [RFC6241], but with additional flexibility in naming the target datastore. If an <edit-data> operation is invoked on a non-writable datastore, then an error is returned, as specified in "ietf-netconf-nmda" (see Section 4).

```
+---x edit-data
   +---w input
      +---w datastore            ds:datastore-ref
      +---w default-operation?   enumeration
      +---w (edit-content)
         +--:(config)
         |  +---w config?        <anydata>
         +--:(url)
            +---w url?           inet:uri {nc:url}?
```

The "datastore" parameter is a datastore identity that indicates the desired target datastore where changes should be made.

The "default-operation" parameter selects the default operation to use. It is a copy of the "default-operation" parameter of the <edit-config> operation.

The "edit-content" parameter specifies the content for the edit operation. It mirrors the "edit-content" choice of the <edit-config> operation. Note, however, that the "config" element in the "edit-content" choice of <edit-data> uses "anydata" (introduced in YANG 1.1) while the "config" element in the "edit-content" choice of <edit-config> used "anyxml".

The <edit-data> operation does not support the "error-option" and the "test-option" parameters that were part of the <edit-config> operation. The error behaviour of <edit-data> corresponds to the "error-option" "rollback-on-error".

If the "with-defaults" capability is supported by the server, the semantics of editing modes is the same as for <edit-config>, as described in section 4.5.2 of [RFC6243].

Semantics for "with-defaults" in <edit-data> operations on any non conventional configuration datastores should be defined by the specification for the datastore.

3.1.2.1. Example: Setting a leaf of an interface in <running>

The following example shows the <edit-data> version of the first <edit-config> example in Section 7.2 of [RFC6241], setting the MTU to 1500 on an interface named "Ethernet0/0" in the running configuration datastore.

```
<rpc message-id="103"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-nmda"
             xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
    <datastore>ds:running</datastore>
    <config>
      <top xmlns="http://example.com/schema/1.2/config">
        <interface>
          <name>Ethernet0/0</name>
          <mtu>1500</mtu>
        </interface>
      </top>
    </config>
  </edit-data>
</rpc>


<rpc-reply message-id="103"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

The other <edit-config> examples shown in Section 7.2 can be translated to <edit-data> examples in a similar way.

3.2. Augmentations to NETCONF Operations

Several of the operations defined in the base NETCONF YANG module "ietf-netconf" [RFC6241] may be used with new datastores. Hence, the <lock>, <unlock>, and <validate> operations are augmented with a new "datastore" leaf that can select the desired datastore. If a <lock>, <unlock>, or <validate> operation is not supported on a particular datastore then an error is returned, as specified in "ietf-netconf-nmda" (see Section 4).

4. NETCONF Datastores YANG Module

This module imports definitions from [RFC6991], [RFC6241], [RFC6243], and [RFC8342].

RFC Ed.: update the date below with the date of RFC publication and remove this note.

```
<CODE BEGINS> file "ietf-netconf-nmda@2018-10-09"

module ietf-netconf-nmda {
  yang-version 1.1;
```

```
     namespace "urn:ietf:params:xml:ns:yang:ietf-netconf-nmda";
     prefix ncds;

     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-datastores {
       prefix ds;
       reference "RFC 8342: Network Management Datastore Architecture.";
     }
     import ietf-origin {
       prefix or;
       reference "RFC 8342: Network Management Datastore Architecture.";
     }
     import ietf-netconf {
       prefix nc;
       reference "RFC 6241: Network Configuration Protocol (NETCONF)";
     }
     import ietf-netconf-with-defaults {
       prefix ncwd;
       reference "RFC 6243: With-defaults Capability for NETCONF.";
     }

     organization
       "IETF NETCONF Working Group";
     contact
       "WG Web:   <https://datatracker.ietf.org/wg/netconf/>

        WG List:  <mailto:netconf@ietf.org>

        Author:   Martin Bjorklund
                  <mailto:mbj@tail-f.com>

        Author:   Juergen Schoenwaelder
                  <mailto:j.schoenwaelder@jacobs-university.de>

        Author:   Phil Shafer
                  <mailto:phil@juniper.net>

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

        Author:   Rob Wilton
```

```
                 <rwilton@cisco.com>";
     description
       "This YANG module defines a set of NETCONF operations to support
        the Network Management Datastore Architecture (NMDA).

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

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

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

     // RFC Ed.: update the date below with the date of RFC publication
     // and remove this note.
     // RFC Ed.: replace XXXX with actual RFC number and remove this
     // note.
     revision 2018-10-09 {
       description
         "Initial revision.";
       reference
         "RFC XXXX: NETCONF Extensions to Support the Network Management
                    Datastore Architecture";
     }

     feature origin {
       description
         "Indicates that the server supports the 'origin' annotation.";
       reference
         "RFC 8342: Network Management Datastore Architecture";
     }

     feature with-defaults {
       description
          "NETCONF :with-defaults capability; If the server advertises
           the :with-defaults capability for a session, then this
           feature must also be enabled for that session.  Otherwise,
           this feature must not be enabled.";
       reference
         "RFC 6243: With-defaults Capability for NETCONF, section 4; and
          RFC XXXX: NETCONF Extensions to Support the Network Management
          Datastore Architecture, section 3.1.1.1.";
```

```
     }

     rpc get-data {
       description
         "Retrieve data from an NMDA datastore. The content returned
          by get-data must satisfy all filters, i.e., the filter
          criteria are logically ANDed.

          Any ancestor nodes (including list keys) of nodes selected by
          the filters are included in the response.

          The 'with-origin' parameter is only valid for an operational
          datastore. If 'with-origin' is used with an invalid datastore,
          then the server MUST return an <rpc-error> element with an
          <error-tag> value of 'invalid-value'.

          The 'with-defaults' parameter only applies to the operational
          datastore if the NETCONF :with-defaults and
          :with-operational-defaults capabilities are both advertised.
          If the 'with-defaults' parameter is present in a request for
          which it is not supported, then the server MUST return an
          <rpc-error> element with an <error-tag> value of
          'invalid-value'.";
       input {
         leaf datastore {
           type ds:datastore-ref;
           mandatory true;
           description
             "Datastore from which to retrieve data.

              If the datastore is not supported by the server, then the
              server MUST return an <rpc-error> element with an
              <error-tag> value of 'invalid-value'.";
         }

         choice filter-spec {
           description
             "The content filter specification for this request.";

           anydata subtree-filter {
             description
               "This parameter identifies the portions of the
                target datastore to retrieve.";
             reference
               "RFC 6241: Network Configuration Protocol, Section 6.";
           }
           leaf xpath-filter {
```

```
             if-feature nc: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, then the get-data
                operation fails.

                The expression is evaluated in the following XPath
                context:

                  o  The set of namespace declarations are those in
                     scope on the 'xpath-filter' leaf element.

                  o  The set of variable bindings is empty.

                  o  The function library is the core function library,
                     and the XPath functions defined in section 10 in
                     RFC 7950.

                  o  The context node is the root node of the target
                     datastore.";
           }
         }

         leaf config-filter {
           type boolean;
           description
             "Filter for nodes with the given value for their
              'config' property.  If this leaf is not present, all
              nodes are selected.

              For example, when this leaf is set to 'true', only 'config
              true' nodes are selected.";
         }
         choice origin-filters {
           when 'derived-from-or-self(datastore, "ds:operational")';
           if-feature origin;
           description
             "Filters based on the 'origin' annotation.";

           leaf-list origin-filter {
             type or:origin-ref;
             description
               "Filter based on the 'origin' annotation.  A node matches
```

```
                the filter if its 'origin' annotation is derived from or
                equal to any of the given filter values.";
           }
           leaf-list negated-origin-filter {
             type or:origin-ref;
             description
               "Filter based on the 'origin' annotation.  A node matches
                the filter if its 'origin' annotation is not derived
                from and not equal to any of the given filter values.";
           }
         }

         leaf max-depth {
           type union {
             type uint16 {
               range "1..65535";
             }
             type enumeration {
               enum "unbounded" {
                 description
                   "All descendant nodes are included.";
               }
             }
           }
           default "unbounded";
           description
             "For each node selected by the filters, this parameter
              selects how many conceptual sub-tree levels should be
              returned in the reply.  If the depth is 1, the reply
              includes just the selected nodes but no children.  If the
              depth is 'unbounded', all descendant nodes are included.";
         }

         leaf with-origin {
           when 'derived-from-or-self(../datastore, "ds:operational")';
           if-feature origin;
           type empty;
           description
             "If this parameter is present, the server will return
              the 'origin' annotation for the nodes that has one.";
         }

         uses ncwd:with-defaults-parameters {
           if-feature with-defaults;
         }
       }

       output {
```

```
          anydata data {
            description
              "Copy of the source datastore subset which matched
               the filter criteria (if any).  An empty data
               container indicates that the request did not
               produce any results.";
          }
        }
      }

      rpc edit-data {
        description
          "Edit data in an NMDA datastore.

           If an error condition occurs such that an error severity
           <rpc-error> element is generated, the server will stop
           processing the <edit-data> operation and restore the
           specified configuration to its complete state at
           the start of this <edit-data> operation.";
        input {
          leaf datastore {
            type ds:datastore-ref;
            mandatory true;
            description
              "Datastore which is the target of the edit-data operation.

               If the target datastore is not writable, or is not
               supported by the server, then the server MUST return an
               <rpc-error> element with an <error-tag> value of
               'invalid-value'.";
          }
          leaf default-operation {
            type enumeration {
              enum "merge" {
                description
                  "The default operation is merge.";
              }
              enum "replace" {
                description
                  "The default operation is replace.";
              }
              enum "none" {
                description
                  "There is no default operation.";
              }
            }
            default "merge";
            description
```

```
             "The default operation to use.";
         }
         choice edit-content {
           mandatory true;
           description
             "The content for the edit operation.";

           anydata config {
             description
               "Inline config content.";
           }
           leaf url {
             if-feature nc:url;
             type inet:uri;
             description
               "URL based config content.";
           }
         }
       }
     }

     /*
      * Augment the lock and unlock operations with a
      * "datastore" parameter.
      */

     augment "/nc:lock/nc:input/nc:target/nc:config-target" {
       description
         "Add NMDA Datastore as target.";
       leaf datastore {
         type ds:datastore-ref;
         description
           "Datastore to lock.

            The lock operation is only supported on writable datastores.

            If the lock operation is not supported by the server on the
            specified target datastore, then the server MUST return an
            <rpc-error> element with an <error-tag> value of
            'invalid-value'.";
       }
     }
     augment "/nc:unlock/nc:input/nc:target/nc:config-target" {
       description
         "Add NMDA Datastore as target.";
       leaf datastore {
         type ds:datastore-ref;
         description
```

```
           "Datastore to unlock.

            The unlock operation is only supported on writable
            datastores.

            If the unlock operation is not supported by the server on
            the specified target datastore, then the server MUST return
            an <rpc-error> element with an <error-tag> value of
            'invalid-value'.";
       }
     }

     /*
      * Augment the validate operation with a
      * "datastore" parameter.
      */

     augment "/nc:validate/nc:input/nc:source/nc:config-source" {
       description
         "Add NMDA Datastore as source.";
       leaf datastore {
         type ds:datastore-ref;
         description
           "Datastore to validate.

            The validate operation is supported only on configuration
            datastores.

            If the validate operation is not supported by the server on
            the specified target datastore, then the server MUST return
            an <rpc-error> element with an <error-tag> value of
            'invalid-value'.";

       }
     }
   }

   <CODE ENDS>
```

5. IANA Considerations

This document registers two capability identifier URNs in the "Network Configuration Protocol (NETCONF) Capability URNs" registry:

```
  Index
  Capability Identifier
  ---------------------
  :yang-library:1.1
  urn:ietf:params:netconf:capability:yang-library:1.1

  :with-operational-defaults
  urn:ietf:params:netconf:capability:with-operational-defaults:1.0
```

This document registers a URI in the "IETF XML Registry" [RFC3688]. Following the format in RFC 3688, the following registration has been made.

URI: urn:ietf:params:xml:ns:yang:ietf-netconf-nmda

Registrant Contact: The IESG.

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

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

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

6. Security Considerations

The YANG module defined in this document extends the base operations of the NETCONF [RFC6241] protocol. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242].

The network configuration access control model [RFC8341] provides the means to restrict access for particular NETCONF users to a preconfigured subset of all available NETCONF protocol operations and content.

The security considerations for the base NETCONF protocol operations (see Section 9 of [RFC6241]) apply to the new NETCONF <get-data> and <edit-data> operations defined in this document.

7. References

7.1. Normative References

[I-D.ietf-netconf-rfc7895bis]
Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", draft-ietf-netconf-rfc7895bis-06 (work in progress), April 2018.

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

[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>.

[RFC6243] Bierman, A. and B. Lengyel, "With-defaults Capability for NETCONF", RFC 6243, DOI 10.17487/RFC6243, June 2011, <https://www.rfc-editor.org/info/rfc6243>.

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

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

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

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

[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>.

7.2. Informative References

[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>.

Authors' Addresses

Martin Bjorklund
Tail-f Systems

Email: mbj@tail-f.com

Juergen Schoenwaelder
Jacobs University

Email: j.schoenwaelder@jacobs-university.de

Phil Shafer
Juniper Networks

Email: phil@juniper.net

Kent Watsen
Juniper Networks

Email: kwatsen@juniper.net

Robert Wilton
Cisco Systems

Email: rwilton@cisco.com

Network Working Group
Internet-Draft
Updates: 8040 (if approved)
Intended status: Standards Track
Expires: April 12, 2019

M. Bjorklund
Tail-f Systems
J. Schoenwaelder
Jacobs University
P. Shafer
K. Watsen
Juniper Networks
R. Wilton
Cisco Systems
October 9, 2018

RESTCONF Extensions to Support the Network Management Datastore Architecture
draft-ietf-netconf-nmda-restconf-05

Abstract

This document extends the RESTCONF protocol defined in RFC 8040 in order to support the Network Management Datastore Architecture defined in RFC 8342.

This document updates RFC 8040 by introducing new datastore resources, adding a new query parameter, and requiring the usage of I-D.ietf-netconf-rfc7895bis by RESTCONF servers implementing the Network Management Datastore Architecture.

RFC Ed.: Please replace "I-D.ietf-netconf-rfc7895bis" above with its final RFC assignment and remove this note.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document extends the RESTCONF protocol defined in [RFC8040] in order to support the Network Management Datastore Architecture (NMDA) defined in [RFC8342].

This document updates [RFC8040] in order to enable RESTCONF clients to discover which datastores are supported by the RESTCONF server, determine which modules are supported in each datastore, and to interact with all the datastores supported by the NMDA. Specifically, the update introduces new datastore resources, adds a new query parameter, and requires the usage of [I-D.ietf-netconf-rfc7895bis] by RESTCONF servers implementing the NMDA.

The solution presented in this document is backwards compatible with [RFC8040]. This is achieved by only adding new resources and leaving the semantics of the existing resources unchanged.

1.1. Terminology

This document uses the terminology defined by the NMDA [RFC8342].

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

2. Datastore and YANG Library Requirements

RFC Ed.: Please update 201X-XX-XX below with correct date and remove this note.

An NMDA-compliant RESTCONF server MUST support the operational state datastore and it MUST implement at least revision 201X-XX-XX of the "ietf-yang-library" module defined in [I-D.ietf-netconf-rfc7895bis].

Such a server identifies that it supports the NMDA both by implementing the {+restconf}/ds/ietf-datastores:operational resource, and by implementing at least revision 201X-XX-XX of the "ietf-yang-library" module.

A RESTCONF client can test if a server supports the NMDA by using either the HEAD or GET methods on {+restconf}/ds/ietf-datastores:operational.

A RESTCONF client can discover which datastores and YANG modules the server supports by reading the YANG library information from the operational state datastore.

3. RESTCONF Extensions

This section describes the RESTCONF extensions needed to support the NMDA.

3.1. New Datastore Resources

This document defines a set of new resources representing datastores as defined in [RFC8342]. These resources are available using the following resource path template:

```
{+restconf}/ds/<datastore>
```

The <datastore> path component is encoded as an "identityref" according to the JSON encoding rules for identities, defined in Section 6.8 of [RFC7951]. The namespace-qualified form MUST be used. Such an identity MUST be derived from the "datastore" identity defined in the "ietf-datastores" YANG module [RFC8342].

Specifically:

- o The resource {+restconf}/ds/ietf-datastores:operational refers to the operational state datastore.
- o The resource {+restconf}/ds/ietf-datastores:running refers to the running configuration datastore.
- o The resource {+restconf}/ds/ietf-datastores:intended refers to the intended configuration datastore.

An NMDA-compliant server MUST implement {+restconf}/ds/ietf-datastores:operational. Other datastore resources MAY be implemented.

YANG actions can only be invoked in {+restconf}/ds/ietf-datastores:operational.

If a server implements other datastores, such as the example datastore "ds-ephemeral" in the module "example-ds-ephemeral", the server would implement the resource {+restconf}/ds/example- ds-ephemeral:ds-ephemeral.

3.2. Protocol Operations

The protocol operations available for the new datastore resources (Section 3.1) are the same as the protocol operations defined in [RFC8040] for the {+restconf}/data resource with the following exceptions:

- o Dynamic configuration datastores are excluded, as each dynamic configuration datastore definition needs to be reviewed for what protocol operations it supports.
- o Some datastores are read-only by nature (e.g., <intended>), and hence any attempt to modify these datastores will fail. A server MUST return a response with a "405 Method Not Allowed" status-line and error-tag value "operation-not-supported".
- o The semantics of the "with-defaults" query parameter ([RFC8040], Section 4.8.9) differs when interacting with the operational state datastore. The semantics are described below, in Section 3.2.1.

- o [RFC8040], Section 3.5.4, paragraph 3 does not apply when interacting with any resource under {+restconf}/ds.

3.2.1. With-defaults query parameter on the operational state datastore

The "with-defaults" query parameter ([RFC8040], Section 4.8.9) is OPTIONAL to support when interacting with {+restconf}/ds/ietf-datastores:operational. The associated capability to indicate a server's support is identified with the URI:

urn:ietf:params:restconf:capability:with-operational-defaults:1.0

For servers that support it, the behavior of the "with-defaults" query parameter on the operational state datastore is defined as follows:

- o If no "with-defaults" query parameter is specified, or if it is set to "explicit", "report-all", or "report-all-tagged", then the "in use" values, as defined in [RFC8342] section 5.3, are returned from the operational state datastore, even if a node happens to have a default statement in the YANG module and this default value is being used by the server. If the "with-defaults" parameter is set to "report-all-tagged", any values that match the schema default are tagged with additional metadata, as described in [RFC8040], Section 4.8.9.

- o If the "with-defaults" query parameter is set to "trim", all "in use" values are returned, except that the output is filtered to exclude any values that match the default defined in the YANG schema.

Servers are not required to support all values in the "with-defaults" query parameter on the operational state datastore. If a request is made using a value that is not supported, then the error handling behavior is as described in ([RFC8040], Section 4.8.9).

3.2.2. New "with-origin" Query Parameter

A new query parameter named "with-origin" is added to the GET operation. If present, it requests that the server includes "origin" metadata annotations in its response, as detailed in the NMDA. This parameter is only valid when querying {+restconf}/ds/ietf-datastores:operational or any datastores with identities derived from the "operational" identity. Otherwise, if an invalid datastore is specified then the server MUST return a response with a "400 Bad Request" status-line, using an error-tag value of "invalid-value". "origin" metadata annotations are not included unless a client explicitly requests them.

Data in the operational state datatstore can come from multiple sources. The server should return the most accurate value for the "origin" metadata annotation as possible, indicating the source of the operational value, as specified in Section 5.3.4 of [RFC8342].

When encoding the origin metadata annotation for a hierarchy of returned nodes, the annotation can be omitted for a child node when the value matches that of the parent node, as described in "ietf-origin" YANG module [RFC8342].

The "with-origin" query parameter is OPTIONAL to support. It is identified with the URI:

```
urn:ietf:params:restconf:capability:with-origin:1.0
```

4. IANA Considerations

This document defines two capability identifier URNs in the "RESTCONF Capability URNs" registry defined in [RFC8040]:

```
Index
Capability Identifier
---------------------

:with-origin
urn:ietf:params:restconf:capability:with-origin:1.0

:with-operational-defaults
urn:ietf:params:restconf:capability:with-operational-defaults:1.0
```

5. Security Considerations

This document extends the RESTCONF protocol by introducing new datastore resources. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446]. The RESTCONF protocol uses the network configuration access control model [RFC8341], which provides the means to restrict access for particular RESTCONF users to a preconfigured subset of all available RESTCONF protocol operations and content.

The security constraints for the base RESTCONF protocol (see Section 12 of [RFC8040]) apply to the new RESTCONF datastore resources defined in this document.

6. Normative References

[I-D.ietf-netconf-rfc7895bis]
Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", draft-ietf-netconf-rfc7895bis-06 (work in progress), April 2018.

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

[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>.

[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>.

Authors' Addresses

Martin Bjorklund
Tail-f Systems

Email: mbj@tail-f.com

Juergen Schoenwaelder
Jacobs University

Email: j.schoenwaelder@jacobs-university.de

Phil Shafer
Juniper Networks

Email: phil@juniper.net

Kent Watsen
Juniper Networks

Email: kwatsen@juniper.net

Robert Wilton
Cisco Systems

Email: rwilton@cisco.com

NETCONF
Internet-Draft
Intended status: Standards Track
Expires: February 16, 2020

E. Voit
Cisco Systems
H. Birkholz
Fraunhofer SIT
A. Bierman
YumaWorks
A. Clemm
Futurewei
T. Jenkins
Cisco Systems

August 15, 2019

Notification Message Headers and Bundles
draft-ietf-netconf-notification-messages-07

Abstract

This document defines a new notification message format. Included are:

- o a new notification mechanism and encoding to replace the one way operation of RFC-5277
- o a set of common, transport agnostic message header objects.
- o how to bundle multiple event records into a single notification message.
- o how to ensure these new capabilities are only used with capable receivers.

Status of This Memo

Table of Contents

1. Introduction

Mechanisms to support subscription to event notifications have been defined in [I-D.draft-ietf-netconf-subscribed-notifications] and [I-D.ietf-netconf-yang-push]. Work on those documents has shown that notifications described in [RFC7950] section 7.16 could benefit from transport independent headers. With such headers, communicating the following information to receiving applications can be done without explicit linkage to an underlying transport protocol:

- o the time the notification was generated
- o the time the notification was placed in a message and queued for transport
- o an identifier for the process generating the notification
- o signatures to verify authenticity
- o a subscription id which allows a notification be correlated with a request for that notification
- o multiple notifications bundled into one transportable message
- o a message-id allowing a receiver to check for message loss/ reordering

The document describes information elements needed for the functions above. It also provides instances of YANG structures [I-D.draft-ietf-netmod-yang-data-ext] for sending messages containing one or more notifications to a receiver.

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 definition of notification is in [RFC7950] Section 4.2.10. Publisher, receiver, subscription, and event occurrence time are defined in [I-D.draft-ietf-netconf-subscribed-notifications].

3. Header Objects

There are a number of transport independent headers which should have common definition. These include:

- o subscription-id: provides a reference into the reason the publisher believed the receiver wishes to be notified of this specific information.
- o notification-time: the origination time where a notification was fully generated within the publisher.

- o notification-id: Identifies an instance of an emitted notification to a receiver.

- o observation-domain-id: identifies the publisher process which discovered and recorded the event notification. (note: look to reuse the domains set up with IPFIX.)

- o message-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 message-id: for a specific message generator, this identifies a message which includes one or more event records. The message-id increments by one with sequential messages.

- o message-generator-id: identifier for the process which created the message. This allows disambiguation of an information source, such as the identification of different line cards sending the messages. Used in conjunction with previous-message-id, this can help find drops and duplications when messages are coming from multiple sources on a device. If there is a message-generator-id in the header, then the previous-message-id MUST be the message-id from the last time that message-generator-id was sent.

4. Encapsulation of Header Objects in Messages

A specific set of well-known objects are of potential use to networking layers prior being interpreted by some receiving application layer process. By exposing this object information as part of a header, and by using standardized object names, it becomes possible for this object information to be leveraged in transit.

The objects defined in the previous section are these well-known header objects. These objects are identified within a dedicated header subtree which leads off a particular transportable message. This allows header objects to be easily be decoupled, stripped, and processed separately.

A receiver which supporting this document MUST be able to handle receipt of either type of message from a publisher.

4.1. One Notification per Message

This section has been deleted from previous versions. It will be re-instated if NETCONF WG members are not comfortable with the efficiency of the solution which can encode many notifications per message, as described below.

4.2. Many Notifications per Message

While possible in some scenarios, it often inefficient to marshal and transport every notification independently. Instead, scale and processing speed can be improved by placing multiple notifications into one transportable bundle.

The format of this bundle appears in the YANG structure below, and is fully defined in the YANG module. There are three parts of this bundle:

- a message header describing the marshaling, including information such as when the marshaling occurred

- a list of encapsulated information

- an optional message footer for whole-message signing and message-generator integrity verification.

Within the list of encapsulated notifications, there are also three parts:

- a notification header defining what is in an encapsulated notification

- the actual notification itself

- an optional notification footer for individual notification signing and observation-domain integrity verification.

```
   structure message
     +--ro message!
        +--ro message-header
        |  +--ro message-time            yang:date-and-time
        |  +--ro message-id?             uint32
        |  +--ro message-generator-id?   string
        |  +--ro notification-count?     uint16
        +--ro notifications*
        |  +--ro notification-header
        |  |  +--ro notification-time         yang:date-and-time
        |  |  +--ro yang-module?              yang:yang-identifier
        |  |  +--ro subscription-id*          uint32
        |  |  +--ro notification-id?          uint32
        |  |  +--ro observation-domain-id?    string
        |  +--ro notification-contents?
        |  +--ro notification-footer!
        |     +--ro signature-algorithm    string
        |     +--ro signature-value        string
        |     +--ro integrity-evidence?    string
        +--ro message-footer!
           +--ro signature-algorithm    string
           +--ro signature-value        string
           +--ro integrity-evidence?    string
```

An XML instance of a message might look like:

```
<structure bundled-message
  xmlns="urn:ietf:params:xml:ns:yang:ietf-notification-messages:1.0">
  <message-header>
    <message-time>
          2017-02-14T00:00:05Z
    </message-time>
    <message-id>
          456
    </message-id>
    <notification-count>
          2
    </notification-count>
  </message-header>
  <notifications>
    <notification>
      <notification-header>
        <notification-time>
            2017-02-14T00:00:02Z
        </notification-time>
        <subscription-id>
            823472
        </subscription-id>
        <yang-module>
            ietf-yang-push
        </yang-module>
        <yang-notification-name>
            push-change-update
        </yang-notification-name>
      </notification-header>
      <notification-contents>
        <push-change-update xmlns=
          "urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
          <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-contents>
    </notification>
    <notification>
          ...(record #2)...
    </notification>
  </notifications>
</structure>
```

5. Configuration of Headers

A publisher MUST select the set of headers to use within any particular message. The two mandatory headers which MUST always be applied are 'message-time' and 'subscription-id'

Beyond these two mandatory headers, additional headers MAY be included. Configuration of what these optional headers should be can come from the following sources:

1. Publisher wide default headers which are placed on all notifications. An optional header is a publisher default if its identity is included within the 'additional-headers' leaf-list.

2. More notification specific headers may also be desired. If new headers are needed for a specific type of YANG notification, these can be populated through 'additional-notification-headers' leaf-list.

3. An application process may also identify common headers to use when transporting notifications for a specific subscription. How such application specific configuration is accomplished within the publisher is out-of-scope.

The set of headers selected and populated for any particular message is derived from the union of the mandatory headers and configured optional headers.

The YANG tree showing elements of configuration is depicted in the following figure.

```
module: ietf-notification-messages
    +--rw additional-default-headers {publisher}?
       +--rw additional-headers*                         optional-header
       +--rw yang-notification-specific-default*
          |                        [yang-module yang-notification-name]
          +--rw yang-module                        yang:yang-identifier
          +--rw yang-notification-name                notification-type
          +--rw additional-notification-headers*
                                           optional-notification-header
```

Configuration Model structure

Of note in this tree is the optional feature of 'publisher'. This feature indicates an ability to send notifications. A publisher supporting this specification MUST also be able to parse any messages received as defined in this document.

6. Discovering Receiver Support

We need capability exchange from the receiver to the publisher at transport session initiation to indicate support for this specification.

For all types of transport connections, if the receiver indicates support for this specification, then it MAY be used. In addition, [RFC5277] one-way notifications MUST NOT be used if the receiver indicates support for this specification to a publisher which also supports it.

Where NETCONF transport is used, advertising this specification's namespace during an earlier client capabilities discovery phase MAY be used to indicate support for this specification:

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>
      urn:ietf:params:xml:ns:yang:ietf-notification-messages:1.0
    </capability>
  </capabilities>
  <session-id>4</session-id>
</hello>
```

NOTE: It is understood that even though it is allowed in [RFC6241] section 8.1, robust NETCONF client driven capabilities exchange is not something which is common in implementation. Therefore reviewers are asked to submit alternative proposals to the mailing list.

For RESTCONF, a mechanism for capability discovery is TBD. Proposals are welcome here.

The mechanism described above assumes that a capability discovery phase happens before a subscription is started. This is not always the case. There is no guarantee that a capability exchange has taken place before the messages are emitted. A solution for this in the case of HTTP based transport could be that a receiver would have to reply "ok" and also return the client capabilities as part a response to the initiation of the POST.

7. YANG Module

```
<CODE BEGINS> file "ietf-notification-messages@2019-08-16.yang"
module ietf-notification-messages {
  yang-version 1.1;
  namespace
```

```
     "urn:ietf:params:xml:ns:yang:ietf-notification-messages";
  prefix nm;

  import ietf-yang-types { prefix yang; }
  import ietf-yang-structure-ext { prefix sx; }


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

     Editor:   Eric Voit
               <mailto:evoit@cisco.com>

     Editor:   Henk Birkholz
               <mailto:henk.birkholz@sit.fraunhofer.de>

     Editor:   Alexander Clemm
               <mailto:ludwig@clemm.org>

     Editor:   Andy Bierman
               <mailto:andy@yumaworks.com>

     Editor:   Tim Jenkins
               <mailto:timjenki@cisco.com>";

  description
    "This module contains conceptual YANG specifications for
    messages carrying notifications with well-known header objects.";

  revision 2019-08-16 {
    description
      "Initial version.";

    reference
      "draft-ietf-netconf-notification-messages-07";
  }

 /*
  * FEATURES
  */

  feature publisher {
    description
      "This feature indicates that support for both publisher and
      receiver of messages complying to the specification.";
  }
```

```
  /*
   * IDENTITIES
   */

   /* Identities for common headers */

  identity common-header {
    description
      "A well-known header which can be included somewhere within a
      message.";
  }

  identity message-time {
    base common-header;
    description
      "Header information consisting of time the message headers were
      generated prior to being sent to transport";
  }

  identity subscription-id {
    base common-header;
    description
      "Header information consisting of the identifier of the
      subscription associated with the notification being
      encapsulated.";
  }

  identity notification-count {
    base common-header;
    description
      "Header information consisting of the quantity of notifications in
      a bundled-message for a specific receiver.";
  }

  identity optional-header {
    base common-header;
    description
      "A well-known header which an application may choose to include
      within a message.";
  }

  identity message-id {
    base optional-header;
    description
      "Header information that identifies a message to a specific
      receiver";
  }
```

```
  identity message-generator-id {
    base optional-header;
    description
      "Header information consisting of an identifier for a software
      entity which created the message (e.g., linecard 1).";
  }

  identity message-signature {
    base optional-header;
    description
      "Identifies two elements of header information consisting of a
      signature and the signature type for the contents of a message.
      Signatures can be useful for originating applications to
      verify record contents even when shipping over unsecure
      transport.";
  }

  identity message-integrity-evidence {
    base optional-header;
    description
      "Header information consisting of the information which backs up
      the assertions made as to the validity of the information
      provided within the message.";
  }


  identity optional-notification-header {
    base optional-header;
    description
      "A well-known header which an application may choose to include
      within a message.";
  }

  identity notification-time  {
    base optional-notification-header;
    description
      "Header information consisting of the time an originating process
      created the notification.";
  }

  identity notification-id {
    base optional-notification-header;
    description
      "Header information consisting of an identifier for an instance
      of a notification. If access control based on a message's receiver may
      strip information from within the notification, this notification-id MUST
      allow the identification of the specific contents of notification as it
      exits the publisher.";
```

```
  }

  identity observation-domain-id {
    base optional-notification-header;
    description
      "Header information identifying the software entity which created
      the notification (e.g., process id).";
  }

  identity notification-signature {
    base optional-notification-header;
    description
      "Header information consisting of a signature which can be used to prove
      the authenticity that some asserter validates over the information
      provided within the notification.";
  }

  identity notification-integrity-evidence {
    base optional-notification-header;
    description
      "Header information consisting of the information which backs up
      the assertions made as to the validity of the information
      provided within the notification.";
  }



  /*
   * TYPEDEFs
   */

  typedef optional-header {
    type identityref {
      base optional-header;
    }
    description
      "Type of header object which may be included somewhere within a
      message.";
  }

  typedef optional-notification-header {
    type identityref {
      base optional-notification-header;
    }
    description
      "Type of header object which may be included somewhere within a
      message.";
  }
```

```
  typedef notification-type {
    type string {
      pattern '[a-zA-Z_][a-zA-Z0-9\-_.]*';
    }
    description
      "The name of a notification within a YANG module.";
    reference
      "RFC-7950 Section 7.16";
  }

  /*
   * GROUPINGS
   */

  grouping message-header {
    description
      "Header information included with a message.";
    leaf message-time {
      type yang:date-and-time;
      mandatory true;
      description
        "Time the message was generated prior to being sent to
        transport.";
    }
    leaf message-id {
      type uint32;
      description
        "Id for a message going to a receiver from a message
        generator.  The id will increment by one with each message sent
        from a particular message generator, allowing the message-id
        to be used as a sequence number.";
    }
    leaf message-generator-id {
      type string;
      description
        "Software entity which created the message (e.g., linecard 1).
         The combination of message-id and message-generator-id must be
         unique until reset or a roll-over occurs.";
    }
    leaf notification-count {
      type uint16;
      description
        "Quantity of notifications in a bundled-message to a
        specific receiver.";
    }
  }

  grouping notification-header {
```

```
    description
      "Common informational objects which might help a receiver
      interpret the meaning, details, or importance of a notification.";
    leaf notification-time {
      type yang:date-and-time;
      mandatory true;
      description
        "Time the system recognized the occurrence of an event.";
    }
    leaf yang-module {
      type yang:yang-identifier;
      description
        "Name of the YANG module supported by the publisher.";
    }
    leaf-list subscription-id {
      type uint32;
      description
        "Id of the subscription which led to the notification being
        generated.";
    }
    leaf notification-id {
      type uint32;
      description
        "Identifier for the notification record.";
    }
    leaf observation-domain-id {
      type string;
      description
        "Software entity which created the notification record (e.g.,
        process id).";
    }
  }

  grouping security-footer {
    description
      "Reusable grouping for common objects which apply to the signing of
      notifications or messages.";
    leaf signature-algorithm {
      type string;
      mandatory true;
      description
        "The technology with which an originator signed of some
        delineated contents.";
    }
    leaf signature-value {
      type string;
      mandatory true;
      description
```

```
        "Any originator signing of the contents of a header and
        content.  This is useful for verifying contents even when
        shipping over unsecure transport.";
    }
    leaf integrity-evidence {
      type string;
      description
        "This mechanism allows a verifier to ensure that the use of the
        private key, represented by the corresponding public key
        certificate, was performed with a TCG compliant TPM
        environment.  This evidence is never included in within any
        signature.";
      reference
        "TCG Infrastructure Workgroup, Subject Key Attestation Evidence
        Extension, Specification Version 1.0, Revision 7.";
    }
  }

  /*
   * YANG encoded structures which can be sent to receivers
   */

   sx:structure message {
    container message {
      presence
        "Indicates attempt to communicate notifications to a receiver.";
      description
        "Message to a receiver containing one or more notifications";
      container message-header {
        description
          "Header info for messages.";
        uses message-header;
      }
      list notifications {
        description
          "Set of notifications to a receiver.";
        container notification-header {
          description
            "Header info for a notification.";
          uses notification-header;
        }
        anydata notification-contents {
          description
            "Encapsulates objects following YANG's notification-stmt
            grammar of RFC-7950 section 14.  Within are the notified
            objects the publisher actually generated in order to be
            passed to a receiver after all filtering has completed.";
        }
```

```
        container notification-footer {
          presence
            "Indicates attempt to secure a notification.";
          description
            "Signature and evidence for messages.";
          uses security-footer;
        }
      }
      container message-footer {
        presence
          "Indicates attempt to secure the entire message.";
        description
          "Signature and evidence for messages.";
        uses security-footer;
      }
    }
  }

  /*
   * DATA-NODES
   */

  container additional-default-headers {
    if-feature "publisher";
    description
      "This container maintains a list of which additional notifications
      should use which optional headers if the receiver supports this
      specification.";
    leaf-list additional-headers {
      type optional-header;
      description
        "This list contains the identities of the optional header types
        which are to be included within each message from this
        publisher.";
    }
    list yang-notification-specific-default {
      key "yang-module yang-notification-name";
      description
        "For any included YANG notifications, this list provides
        additional optional headers which should be placed within the
        container notification-header if the receiver supports this
        specification.  This list incrementally adds to any headers
        indicated within the leaf-list 'additional-headers'.";
      leaf yang-module {
        type yang:yang-identifier;
        description
          "Name of the YANG module supported by the publisher.";
      }
```

```
      leaf yang-notification-name {
        type notification-type;
         description
           "The name of a notification within a YANG module.";
      }
      leaf-list additional-notification-headers {
         type optional-notification-header;
         description
           "The set of additional default headers which will be sent
           for a specific notification.";
      }
    }
  }
}

<CODE ENDS>
```

8. Backwards Compatibility

With this specification, there is no change to YANG's 'notification' statement

Legacy clients are unaffected, and existing users of [RFC5277], [RFC7950], and [RFC8040] are free to use current behaviors until all involved device support this specification.

9. Security Considerations

Certain headers might be computationally complex for a publisher to deliver. Signatures or encryption are two examples of this. It MUST be possible to suspend or terminate a subscription due to lack of resources based on this reason.

Decisions on whether to bundle or not to a receiver are fully under the purview of the Publisher. A receiver could slow delivery to existing subscriptions by creating new ones. (Which would result in the publisher going into a bundling mode.)

10. Acknowledgments

For their valuable comments, discussions, and feedback, we wish to acknowledge Martin Bjorklund, Einar Nilsen-Nygaard, and Kent Watsen.

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-16 (work in progress), August 2019.

[I-D.draft-ietf-netmod-yang-data-ext]
Bierman, A., Bjorklund, M., and K. Watsen, "YANG Data Structure Extensions", draft-ietf-netmod-yang-data-ext (work in progress), July 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>.

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

[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

[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", August 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", August 2019, <https://datatracker.ietf.org/doc/draft-ietf-netconf-yang-push/>.

[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>.

[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>.

Appendix A. Changes between revisions

(To be removed by RFC editor prior to publication)

v06 - v07

- o Updated author affiliation.

v05 - v06

- o With SN and YP getting RFC numbers, revisiting this document.
- o Changed yang-data to draft-ietf-netmod-yang-data-ext's 'structure'.
- o Removed the ability to reference structures other than YANG notifications.

v04 - v05

- o Revision before expiration. Awaiting closure of SN and YP prior to update.

v03 - v04

- o Terminology tweaks.
- o Revision before expiration. Awaiting closure of SN prior to update.

v02 - v03

- o Removed the option for an unbundled message. This might be re-added later for transport efficiency if desired by the WG
- o New message structure driven by the desire to put the signature information at the end.

v01 - v02

- o Fixed the yang-data encapsulation container issue
- o Updated object definitions to point to RFC-7950 definitions
- o Added headers for module and notification-type.

v00 - v01

- o Alternative to 5277 one-way notification added
- o Storage of default headers by notification type
- o Backwards compatibility
- o Capability discovery
- o Move to yang-data
- o Removed dscp and record-type as common headers. (Record type can be determined by the namespace of the record contents. Dscp is useful where applications need internal communications within a Publisher, but it is unclear as to whether this use case need be exposed to a receiver.

Appendix B. Issues being worked

(To be removed by RFC editor prior to publication)

A complete JSON document is supposed to be sent as part of Media Type "application/yang-data+json". As we are sending separate notifications after each other, we need to choose whether we start with some extra encapsulation for the very first message pushed, or if we want a new Media Type for streaming updates.

Improved discovery mechanisms for NETCONF

Need to ensure the proper references exist to a notification definition driven by RFC-7950 which is acceptable to other eventual users of this specification.

Appendix C. Subscription Specific Headers

(To be removed by RFC editor prior to publication)

This section discusses a future functional addition which could leverage this draft. It is included for informational purposes only.

A dynamic subscriber might want to mandate that certain headers be used for push updates from a publisher. Some examples of this include a subscriber requesting to:

- o establish this subscription, but just if transport messages containing the pushed data will be encrypted,
- o establish this subscription, but only if you can attest to the information being delivered in requested notification records, or
- o provide a sequence-id for all messages to this receiver (in order to check for loss).

Providing this type of functionality would necessitate a new revision of the [I-D.draft-ietf-netconf-subscribed-notifications]'s RPCs and state change notifications. Subscription specific header information would overwrite the default headers identified in this document.

Appendix D. Implications to Existing RFCs

(To be removed by RFC editor prior to publication)

YANG one-way exchanges currently use a non-extensible header and encoding defined in section 4 of RFC-5277. These RFCs MUST be updated to enable this draft. These RFCs SHOULD be updated to provide examples

D.1. Implications to RFC-7950

Sections which expose netconf:capability:notification:1.0 are 4.2.10

Sections which provide examples using netconf:notification:1.0 are 7.10.4, 7.16.3, and 9.9.6

D.2. Implications to RFC-8040

Section 6.4 demands use of RFC-5277's netconf:notification:1.0, and later in the section provides an example.

Authors' Addresses

Eric Voit
Cisco Systems

Email: evoit@cisco.com

Henk Birkholz
Fraunhofer SIT

Email: henk.birkholz@sit.fraunhofer.de

Andy Bierman
YumaWorks

Email: andy@yumaworks.com

Alexander Clemm
Futurewei

Email: ludwig@clemm.org

Tim Jenkins
Cisco Systems

Email: timjenki@cisco.com

NETCONF Working Group   K. Watsen
Internet-Draft   Watsen Networks
Intended status: Standards Track   July 2, 2019
Expires: January 3, 2020

RESTCONF Client and Server Models
draft-ietf-netconf-restconf-client-server-14

Abstract

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

Editorial Note (To be removed by RFC Editor)

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

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

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

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

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

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

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

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

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

o Appendix B. Change Log

Status of This Memo

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

Copyright Notice

Table of Contents

1. Introduction

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

1.1. Terminology

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

2. The RESTCONF Client Model

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

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

2.1. Tree Diagram

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

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

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

```
   module: ietf-restconf-client
     +--rw restconf-client
        +---u restconf-client-grouping

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

2.2. Example Usage

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

This example is consistent with the examples presented in Section 2 of [I-D.ietf-netconf-trust-anchors] and Section 3.2 of [I-D.ietf-netconf-keystore].


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

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

  <!-- RESTCONF servers to initiate connections to -->
  <initiate>
    <restconf-server>
      <name>corp-fw1</name>
      <endpoints>
        <endpoint>
          <name>corp-fw1.example.com</name>
          <https>
            <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>
                <local-definition>
                  <algorithm>rsa2048</algorithm>
                  <private-key>base64encodedvalue==</private-key>
                  <public-key>base64encodedvalue==</public-key>
                  <cert>base64encodedvalue==</cert>
                </local-definition>
              </client-identity>
              <server-authentication>
                <ca-certs>explicitly-trusted-server-ca-certs</ca-cer\
ts>
                <server-certs>explicitly-trusted-server-certs</serve\
r-certs>
              </server-authentication>
              <keepalives>
                <max-wait>30</max-wait>
                <max-attempts>3</max-attempts>
```

```
                </keepalives>
              </tls-client-parameters>
              <http-client-parameters>
                <protocol-version>HTTP/1.1</protocol-version>
                <client-identity>
                  <basic>
                    <user-id>bob</user-id>
                    <password>secret</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>
                  <local-definition>
                    <algorithm>rsa2048</algorithm>
                    <private-key>base64encodedvalue==</private-key>
                    <public-key>base64encodedvalue==</public-key>
                    <cert>base64encodedvalue==</cert>
                  </local-definition>
                </client-identity>
                <server-authentication>
                  <ca-certs>explicitly-trusted-server-ca-certs</ca-cer\
   ts>
                  <server-certs>explicitly-trusted-server-certs</serve\
   r-certs>
                </server-authentication>
                <keepalives>
                  <max-wait>30</max-wait>
                  <max-attempts>3</max-attempts>
                </keepalives>
              </tls-client-parameters>
              <http-client-parameters>
                <protocol-version>HTTP/1.1</protocol-version>
                <client-identity>
                  <basic>
```

```
                     <user-id>bob</user-id>
                     <password>secret</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>
               <local-definition>
                 <algorithm>rsa2048</algorithm>
                 <private-key>base64encodedvalue==</private-key>
                 <public-key>base64encodedvalue==</public-key>
                 <cert>base64encodedvalue==</cert>
               </local-definition>
             </client-identity>
             <server-authentication>
               <ca-certs>explicitly-trusted-server-ca-certs</ca-certs>
               <server-certs>explicitly-trusted-server-certs</server-ce\
rts>
             </server-authentication>
           </tls-client-parameters>
           <http-client-parameters>
             <protocol-version>HTTP/1.1</protocol-version>
             <client-identity>
               <basic>
                 <user-id>bob</user-id>
                 <password>secret</password>
               </basic>
             </client-identity>
           </http-client-parameters>
         </https>
       </endpoint>
```

```
     </listen>
   </restconf-client>
```

2.3. YANG Module

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

```
<CODE BEGINS> file "ietf-restconf-client@2019-07-02.yang"
module ietf-restconf-client {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-restconf-client";
  prefix rcc;

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

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

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

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

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

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

```
     contact
       "WG Web:   <http://datatracker.ietf.org/wg/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) 2019 IETF Trust and the persons identified
        as authors of the code. All rights reserved.

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

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

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

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

     // Features

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

```
     }

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

     // Groupings

     grouping restconf-client-grouping {
       description
         "Top-level grouping for RESTCONF client configuration.";
       container initiate {
         if-feature "https-initiate";
         presence "Enables client to initiate TCP connections";
         description
           "Configures client initiating underlying TCP connections.";
         list restconf-server {
           key "name";
           min-elements 1;
           description
             "List of RESTCONF servers the RESTCONF client is to
              initiate connections to in parallel.";
           leaf name {
             type string;
             description
               "An arbitrary name for the RESTCONF server.";
           }
           container endpoints {
             description
               "Container for the list of endpoints.";
             list endpoint {
               key "name";
               min-elements 1;
               ordered-by user;
               description
                 "A non-empty user-ordered list of endpoints for this
                  RESTCONF client to try to connect to in sequence.
                  Defining more than one enables high-availability.";
               leaf name {
                 type string;
                 description
                   "An arbitrary name for this endpoint.";
```

```
               }
               choice transport {
                 mandatory true;
                 description
                   "Selects between available transports. This is a
                    'choice' statement so as to support additional
                    transport options to be augmented in.";
                 case https {
                   if-feature "https-initiate";
                   container https {
                     description
                       "Specifies HTTPS-specific transport
                        configuration.";
                     container tcp-client-parameters {
                       description
                         "A wrapper around the TCP client parameters
                          to avoid name collisions.";
                       uses tcpc:tcp-client-grouping {
                         refine "remote-port" {
                           default "443";
                           description
                             "The RESTCONF client will attempt to
                              connect to the IANA-assigned well-known
                              port value for 'https' (443) if no value
                              is specified.";
                         }
                       }
                     }
                     container tls-client-parameters {
                       must "client-identity" {
                         description
                           "NETCONF/TLS clients MUST pass some
                            authentication credentials.";
                       }
                       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;
                     }
                   }
                 } // https
               } // transport
```

```
          } // endpoint
        } // endpoints
        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).";
            }
          } // reconnect-strategy
        } // restconf-server
      } // initiate

      container listen {
        if-feature "https-listen";
        presence "Enables client to accept call-home connections";
```

```
         description
           "Configures client accepting call-home TCP connections.";
         leaf idle-timeout {
           type uint16;
           units "seconds";
           default 3600; // one hour
           description
             "Specifies the maximum number of seconds that an
              underlying TCP session may remain idle. A TCP session
              will be dropped if it is idle for an interval longer
              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.";
           }
           choice transport {
             mandatory true;
             description
               "Selects between available transports. This is a
                'choice' statement so as to support additional
                transport options to be augmented in.";
             case https {
               if-feature "https-listen";
               container https {
                 description
                   "HTTPS-specific listening configuration for inbound
                    connections.";
                 container tcp-server-parameters {
                   description
                     "A wrapper around the TCP client parameters
                      to avoid name collisions.";
                   uses tcps:tcp-server-grouping {
                     refine "local-port" {
                       default "4336";
                       description
                         "The RESTCONF client will listen on the IANA-
                          assigned well-known port for 'restconf-ch-tls'
                          (4336) if no value is specified.";
```

```
                     }
                   }
                 }
                 container 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 http-client-parameters {
                   description
                     "A wrapper around the HTTP client parameters
                      to avoid name collisions.";
                   uses httpc:http-client-grouping;
                 }
               }
             } // case https
           } // transport
         } // endpoint
       } // listen
     } // restconf-client

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

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

3. The RESTCONF Server Model

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

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

3.1. Tree Diagram

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

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

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

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

  grouping restconf-server-grouping
    +-- client-identification
       +-- cert-maps
          +---u x509c2n:cert-to-name
  grouping restconf-server-listen-stack-grouping
    +-- (transport)
       +--:(http) {http-listen}?
       |  +-- http
       |     +-- external-endpoint
       |     |  +-- address    inet:ip-address
       |     |  +-- port?      inet:port-number
       |     +-- tcp-server-parameters
       |     |  +---u tcps:tcp-server-grouping
       |     +-- http-server-parameters
       |     |  +---u https:http-server-grouping
       |     +-- restconf-server-parameters
       |        +---u rcs:restconf-server-grouping
       +--:(https) {https-listen}?
          +-- https
             +-- tcp-server-parameters
             |  +---u tcps:tcp-server-grouping
             +-- tls-server-parameters
             |  +---u tlss:tls-server-grouping
             +-- http-server-parameters
             |  +---u https:http-server-grouping
             +-- restconf-server-parameters
                +---u rcs:restconf-server-grouping
  grouping restconf-server-callhome-stack-grouping
    +-- (transport)
       +--:(https) {https-listen}?
          +-- https
             +-- tcp-client-parameters
```

```
                |  +---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
     grouping restconf-server-app-grouping
       +-- listen! {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
```

3.2. Example Usage

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

This example is consistent with the examples presented in Section 2 of [I-D.ietf-netconf-trust-anchors] and Section 3.2 of [I-D.ietf-netconf-keystore].

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

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

```
    <!-- endpoints to listen for RESTCONF connections on -->
    <listen>
      <endpoint>
        <name>netconf/tls</name>
        <https>
          <tcp-server-parameters>
            <local-address>11.22.33.44</local-address>
          </tcp-server-parameters>
          <tls-server-parameters>
            <server-identity>
              <local-definition>
                <algorithm>rsa2048</algorithm>
                <private-key>base64encodedvalue==</private-key>
                <public-key>base64encodedvalue==</public-key>
                <cert>base64encodedvalue==</cert>
              </local-definition>
            </server-identity>
            <client-authentication>
              <required/>
              <ca-certs>explicitly-trusted-client-ca-certs</ca-certs>
              <client-certs>explicitly-trusted-client-certs</client-ce\
rts>
            </client-authentication>
          </tls-server-parameters>
          <http-server-parameters>
            <server-name>foo.example.com</server-name>
            <protocol-versions>
              <protocol-version>HTTP/1.1</protocol-version>
              <protocol-version>HTTP/2.0</protocol-version>
            </protocol-versions>
          </http-server-parameters>
          <restconf-server-parameters>
            <client-identification>
              <cert-maps>
                <cert-to-name>
                  <id>1</id>
                  <fingerprint>11:0A:05:11:00</fingerprint>
                  <map-type>x509c2n:san-any</map-type>
                </cert-to-name>
                <cert-to-name>
                  <id>2</id>
                  <fingerprint>B3:4F:A1:8C:54</fingerprint>
                  <map-type>x509c2n:specified</map-type>
                  <name>scooby-doo</name>
                </cert-to-name>
              </cert-maps>
            </client-identification>
          </restconf-server-parameters>
```

```
        </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>
              </tcp-client-parameters>
              <tls-server-parameters>
                <server-identity>
                  <local-definition>
                    <algorithm>rsa2048</algorithm>
                    <private-key>base64encodedvalue==</private-key>
                    <public-key>base64encodedvalue==</public-key>
                    <cert>base64encodedvalue==</cert>
                  </local-definition>
                </server-identity>
                <client-authentication>
                  <required/>
                  <ca-certs>explicitly-trusted-client-ca-certs</ca-cer\
ts>
                  <client-certs>explicitly-trusted-client-certs</clien\
t-certs>
                </client-authentication>
              </tls-server-parameters>
              <http-server-parameters>
                <server-name>foo.example.com</server-name>
                <protocol-versions>
                  <protocol-version>HTTP/1.1</protocol-version>
                  <protocol-version>HTTP/2.0</protocol-version>
                </protocol-versions>
              </http-server-parameters>
              <restconf-server-parameters>
                <client-identification>
                  <cert-maps>
                    <cert-to-name>
                      <id>1</id>
                      <fingerprint>11:0A:05:11:00</fingerprint>
                      <map-type>x509c2n:san-any</map-type>
                    </cert-to-name>
                    <cert-to-name>
```

```
                       <id>2</id>
                       <fingerprint>B3:4F:A1:8C:54</fingerprint>
                       <map-type>x509c2n:specified</map-type>
                       <name>scooby-doo</name>
                     </cert-to-name>
                   </cert-maps>
                 </client-identification>
               </restconf-server-parameters>
             </https>
           </endpoint>
           <endpoint>
             <name>west-data-center</name>
             <https>
               <tcp-client-parameters>
                 <remote-address>west.example.com</remote-address>
               </tcp-client-parameters>
               <tls-server-parameters>
                 <server-identity>
                   <local-definition>
                     <algorithm>rsa2048</algorithm>
                     <private-key>base64encodedvalue==</private-key>
                     <public-key>base64encodedvalue==</public-key>
                     <cert>base64encodedvalue==</cert>
                   </local-definition>
                 </server-identity>
                 <client-authentication>
                   <required/>
                   <ca-certs>explicitly-trusted-client-ca-certs</ca-cer\
   ts>
                   <client-certs>explicitly-trusted-client-certs</clien\
   t-certs>
                 </client-authentication>
               </tls-server-parameters>
               <http-server-parameters>
                 <server-name>foo.example.com</server-name>
                 <protocol-versions>
                   <protocol-version>HTTP/1.1</protocol-version>
                   <protocol-version>HTTP/2.0</protocol-version>
                 </protocol-versions>
               </http-server-parameters>
               <restconf-server-parameters>
                 <client-identification>
                   <cert-maps>
                     <cert-to-name>
                       <id>1</id>
                       <fingerprint>11:0A:05:11:00</fingerprint>
                       <map-type>x509c2n:san-any</map-type>
                     </cert-to-name>
```

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


3.3. YANG Module

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

```
<CODE BEGINS> file "ietf-restconf-server@2019-07-02.yang"
module ietf-restconf-server {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-restconf-server";
  prefix rcs;

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

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

```
         "RFC 6991: Common YANG Data Types";
     }

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

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

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

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

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

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

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

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

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

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

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

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

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

     // Features

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

     feature https-listen {
       description
         "The 'https-listen' feature indicates that the RESTCONF server
          supports opening a port to listen for incoming RESTCONF over
          TLS client connections, whereby the TLS connections are
          terminated by the server itself/";
       reference
         "RFC 8040: RESTCONF Protocol";
```

```
     }

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


     // Groupings

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

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

       container client-identification {  // FIXME: if-feature?
         description
           "Specifies a mapping through which clients MAY be identified
            (i.e., the RESTCONF username) from a supplied certificate.
            Note that a client MAY alternatively be identified via an
            HTTP-level authentication schema.  This configuration does
            not necessitate clients send a certificate (that can be
            controlled via the ietf-restconf-server module).";
         container cert-maps {
           uses x509c2n:cert-to-name;
           description
             "The cert-maps container is used by TLS-based RESTCONF
              servers (even if the TLS sessions are terminated
              externally) to map the RESTCONF client's presented
              X.509 certificate to a RESTCONF username.  If no
              matching and valid cert-to-name list entry can be
              found, then the RESTCONF server MUST close the
              connection, and MUST NOT accept RESTCONF messages
              over it.";
           reference
```

```
             "RFC 7407: A YANG Data Model for SNMP Configuration.";
         }
       }
     }


     grouping restconf-server-listen-stack-grouping {
       description
         "A reusable grouping for configuring a RESTCONF server
          'listen' protocol stack, for a single connection.";
       choice transport {
         mandatory true;
         description
           "Selects between available transports. This is a
            'choice' statement so as to support additional
            transport options to be augmented in.";
         case http {
           if-feature "http-listen";
           container http {
             description
               "Configures RESTCONF server stack assuming that
                TLS-termination is handled externally.";
             container external-endpoint {
               description
                 "Identifies contact information for the external
                  system that terminates connections before passing
                  them thru to this server (e.g., a network address
                  translator or a load balancer).  These values have
                  no effect on the local operation of this server, but
                  may be used by the application when needing to
                  inform other systems how to contact this server.";
               leaf address {
                 type inet:ip-address;
                 mandatory true;
                 description
                   "The IP address or hostname of the external system
                    that terminates incoming RESTCONF client
                    connections before forwarding them to this
                    server.";
               }
               leaf port {
                 type inet:port-number;
                 default "443";
                 description
                   "The port number that the external system listens
                    on for incoming RESTCONF client connections that
                    are forwarded to this server.  The default HTTPS
                    port (443) is used, as expected for a RESTCONF
```

```
                    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 and for many connections.";
     container listen {
       if-feature "https-listen";
       presence
         "Enables the RESTCONF server to listen for RESTCONF
          client connections.";
       description "Configures listen behavior";
       list endpoint {
         key "name";
         min-elements 1;
         description
```

```
            "List of endpoints to listen for RESTCONF connections.";
          leaf name {
            type string;
            description
              "An arbitrary name for the RESTCONF listen endpoint.";
          }
          uses restconf-server-listen-stack-grouping;
        }
      }
      container call-home {
        if-feature "https-call-home";
        presence
          "Enables the RESTCONF server to initiate the underlying
           transport connection to RESTCONF clients.";
        description "Configures call-home behavior";
        list restconf-client {
          key "name";
          min-elements 1;
          description
            "List of RESTCONF clients the RESTCONF server is to
             initiate call-home connections to in parallel.";
          leaf name {
            type string;
            description
              "An arbitrary name for the remote RESTCONF client.";
          }
          container endpoints {
            description
              "Container for the list of endpoints.";
            list endpoint {
              key "name";
              min-elements 1;
              ordered-by user;
              description
                "User-ordered list of endpoints for this RESTCONF
                 client.  Defining more than one enables high-
                 availability.";
              leaf name {
                type string;
                description
                  "An arbitrary name for this endpoint.";
              }
              uses restconf-server-callhome-stack-grouping;
            }
          }
          container connection-type {
            description
              "Indicates the RESTCONF server's preference for how the
```

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

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

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

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

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

                    leaf period {
                      type uint16;
                      units "minutes";
                      default "60";
```

```
                  description
                    "Duration of time between periodic connections.";
                }
                leaf anchor-time {
                  type yang:date-and-time {
                    // constrained to minute-level granularity
                    pattern '\d{4}-\d{2}-\d{2}T\d{2}:\d{2}'
                          + '(Z|[\+\-]\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

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

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

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

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

None of the subtrees or data nodes in the modules defined in this document need to be protected from write operations.

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

None of the subtrees or data nodes in the modules defined in this document need to be protected from read operations.

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

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

5. IANA Considerations

5.1. The IETF XML Registry

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

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

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

5.2. The YANG Module Names Registry

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

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

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

6. References

6.1. Normative References

[I-D.ietf-netconf-keystore]
Watsen, K., "A YANG Data Model for a Keystore", draft-ietf-netconf-keystore-11 (work in progress), June 2019.

[I-D.ietf-netconf-tls-client-server]
Watsen, K., Wu, G., and L. Xia, "YANG Groupings for TLS Clients and TLS Servers", draft-ietf-netconf-tls-client-server-13 (work in progress), June 2019.

[I-D.kwatsen-netconf-http-client-server]
Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", draft-kwatsen-netconf-http-client-server-03 (work in progress), June 2019.

[I-D.kwatsen-netconf-tcp-client-server]
Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", draft-kwatsen-netconf-tcp-client-server-02 (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>.

[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-trust-anchors]
Watsen, K., "A YANG Data Model for a Truststore", draft-ietf-netconf-trust-anchors-05 (work in progress), June 2019.

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

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

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

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

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

Appendix A. Expanded Tree Diagrams

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

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

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

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

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

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

```
        |     |              |  |     |     +--rw public-key
        |     |              |  |     |     |       binary
        |     |              |  |     |     +--rw (private-key-type)
        |     |              |  |     |     |  +--:(private-key)
        |     |              |  |     |     |  |  +--rw private-key?
        |     |              |  |     |     |  |          binary
        |     |              |  |     |     |  +--:(hidden-private-key)
        |     |              |  |     |     |  |  +--rw hidden-private-\
   \key?
        |     |              |  |     |     |  |          empty
        |     |              |  |     |     |  +--:(encrypted-private-k\
   \ey)
        |     |              |  |     |     |     +--rw encrypted-priva\
   \te-key
        |     |              |  |     |     |        +--rw (key-type)
        |     |              |  |     |     |        |  +--:(symmetric-\
   \key-ref)
        |     |              |  |     |     |        |  |  +--rw symmet\
   \ric-key-ref?    leafref
        |     |              |  |     |     |        |  |          {key\
   \store-supported}?
        |     |              |  |     |     |        |  +--:(asymmetric\
   \-key-ref)
        |     |              |  |     |     |        |     +--rw asymme\
   \tric-key-ref?   leafref
        |     |              |  |     |     |        |             {key\
   \store-supported}?
        |     |              |  |     |     |        +--rw value?
        |     |              |  |     |     |                binary
        |     |              |  |     |     +--rw cert?
        |     |              |  |     |     |       end-entity-cert-cms
        |     |              |  |     |     +---n certificate-expiration
        |     |              |  |     |     |  +-- expiration-date
        |     |              |  |     |     |          yang:date-and-ti\
   \me
        |     |              |  |     |     +---x generate-certificate-\
   \signing-request
        |     |              |  |     |        +---w input
        |     |              |  |     |        |  +---w subject
        |     |              |  |     |        |  |       binary
        |     |              |  |     |        |  +---w attributes?
        |     |              |  |     |        |          binary
        |     |              |  |     |        +--ro output
        |     |              |  |     |           +--ro certificate-sig\
   \ning-request
        |     |              |  |     |                   binary
        |     |              |  |     +--:(keystore)
        |     |              |  |              {keystore-supported}?
```

```
     |     |              | |        +--rw keystore-reference
     |     |              | |           +--rw asymmetric-key?
     |     |              | |           |       ks:asymmetric-key-r\
  \ef
     |     |              | |           +--rw certificate?      lea\
  \fref
     |     |              | +--rw server-authentication
     |     |              | |  +--rw ca-certs?
     |     |              | |  |       ts:certificates-ref
     |     |              | |  |       {ts:x509-certificates}?
     |     |              | |  +--rw server-certs?
     |     |              | |          ts:certificates-ref
     |     |              | |          {ts:x509-certificates}?
     |     |              | +--rw hello-params
     |     |              | |       {tls-client-hello-params-config\
  \}?
     |     |              | |  +--rw tls-versions
     |     |              | |  |  +--rw tls-version*   identityref
     |     |              | |  +--rw cipher-suites
     |     |              | |     +--rw cipher-suite*   identityref
     |     |              | +--rw keepalives!
     |     |              |         {tls-client-keepalives}?
     |     |              |    +--rw max-wait?       uint16
     |     |              |    +--rw max-attempts?   uint8
     |     |              +--rw http-client-parameters
     |     |                 +--rw protocol-version?   enumeration
     |     |                 +--rw client-identity
     |     |                 |  +--rw (auth-type)?
     |     |                 |     +--:(basic)
     |     |                 |     |  +--rw basic {basic-auth}?
     |     |                 |     |     +--rw user-id?    string
     |     |                 |     |     +--rw password?   string
     |     |                 |     +--:(bearer)
     |     |                 |     |  +--rw bearer {bearer-auth}?
     |     |                 |     |     +--rw token?   string
     |     |                 |     +--:(digest)
     |     |                 |     |  +--rw digest {digest-auth}?
     |     |                 |     |     +--rw username?   string
     |     |                 |     |     +--rw password?   string
     |     |                 |     +--:(hoba)
     |     |                 |     |  +--rw hoba {hoba-auth}?
     |     |                 |     +--:(mutual)
     |     |                 |     |  +--rw mutual {mutual-auth}?
     |     |                 |     +--:(negotiate)
     |     |                 |     |  +--rw negotiate
     |     |                 |     |          {negotiate-auth}?
     |     |                 |     +--:(oauth)
     |     |                 |     |  +--rw oauth {oauth-auth}?
```

```
        |     |                 |     +--:(scram-sha-1)
        |     |                 |     |  +--rw scram-sha-1
        |     |                 |     |          {scram-sha-1-auth}?
        |     |                 |     +--:(scram-sha-256)
        |     |                 |     |  +--rw scram-sha-256
        |     |                 |     |          {scram-sha-256-auth}?
        |     |                 |     +--:(vapid)
        |     |                 |        +--rw vapid {vapid-auth}?
        |     |                 +--rw proxy-server! {proxy-connect}?
        |     |                    +--rw tcp-client-parameters
        |     |                    |  +--rw remote-address    inet:host
        |     |                    |  +--rw remote-port?
        |     |                    |  |       inet:port-number
        |     |                    |  +--rw local-address?
        |     |                    |  |       inet:ip-address
        |     |                    |  |       {local-binding-supported}?
        |     |                    |  +--rw local-port?
        |     |                    |  |       inet:port-number
        |     |                    |  |       {local-binding-supported}?
        |     |                    |  +--rw keepalives!
        |     |                    |          {keepalives-supported}?
        |     |                    |     +--rw idle-time         uint16
        |     |                    |     +--rw max-probes        uint16
        |     |                    |     +--rw probe-interval    uint16
        |     |                    +--rw tls-client-parameters
        |     |                    |  +--rw client-identity
        |     |                    |  |  +--rw (local-or-keystore)
        |     |                    |  |     +--:(local)
        |     |                    |  |     |        {local-definitions\
   \-supported}?
        |     |                    |  |     |  +--rw local-definition
        |     |                    |  |     |     +--rw algorithm
        |     |                    |  |     |     |       asymmetric-ke\
   \y-algorithm-t
        |     |                    |  |     |     +--rw public-key
        |     |                    |  |     |     |       binary
        |     |                    |  |     |     +--rw (private-key-ty\
   \pe)
        |     |                    |  |     |     |  +--:(private-key)
        |     |                    |  |     |     |  |  +--rw private-k\
   \ey?
        |     |                    |  |     |     |  |          binary
        |     |                    |  |     |     |  +--:(hidden-privat\
   \e-key)
        |     |                    |  |     |     |  |  +--rw hidden-pr\
   \ivate-key?
        |     |                    |  |     |     |  |          empty
        |     |                    |  |     |     |  +--:(encrypted-pri\
```

```
   \vate-key)
        |     |                    |  |     |     |     +--rw encrypted\
   \-private-key
        |     |                    |  |     |     |        +--rw (key-t\
   \ype)
        |     |                    |  |     |     |        |  +--:(symm\
   \etric-key-ref)
        |     |                    |  |     |     |        |  |  +--rw \
   \symmetric-key-ref?    leafref
        |     |                    |  |     |     |        |  |        \
   \  {keystore-supported}?
        |     |                    |  |     |     |        |  +--:(asym\
   \metric-key-ref)
        |     |                    |  |     |     |        |     +--rw \
   \asymmetric-key-ref?   leafref
        |     |                    |  |     |     |        |           \
   \  {keystore-supported}?
        |     |                    |  |     |     |        +--rw value?
        |     |                    |  |     |     |                bina\
   \ry
        |     |                    |  |     |     +--rw cert?
        |     |                    |  |     |     |       end-entity-ce\
   \rt-cms
        |     |                    |  |     |     +---n certificate-exp\
   \iration
        |     |                    |  |     |     |  +-- expiration-date
        |     |                    |  |     |     |          yang:date-\
   \and-time
        |     |                    |  |     |     +---x generate-certif\
   \icate-signing-request
        |     |                    |  |     |        +---w input
        |     |                    |  |     |        |  +---w subject
        |     |                    |  |     |        |  |       binary
        |     |                    |  |     |        |  +---w attribute\
   \s?
        |     |                    |  |     |        |          binary
        |     |                    |  |     |        +--ro output
        |     |                    |  |     |           +--ro certifica\
   \te-signing-request
        |     |                    |  |     |                   binary
        |     |                    |  |     +--:(keystore)
        |     |                    |  |              {keystore-supporte\
   \d}?
        |     |                    |  |        +--rw keystore-reference
        |     |                    |  |           +--rw asymmetric-key?
        |     |                    |  |           |       ks:asymmetric\
   \-key-ref
        |     |                    |  |           +--rw certificate?   \
```

```
   \   leafref
        |     |                    |  +--rw server-authentication
        |     |                    |  |  +--rw ca-certs?
        |     |                    |  |  |       ts:certificates-ref
        |     |                    |  |  |       {ts:x509-certificates}?
        |     |                    |  |  +--rw server-certs?
        |     |                    |  |          ts:certificates-ref
        |     |                    |  |          {ts:x509-certificates}?
        |     |                    |  +--rw hello-params
        |     |                    |  |       {tls-client-hello-params-\
   \config}?
        |     |                    |  |  +--rw tls-versions
        |     |                    |  |  |  +--rw tls-version*
        |     |                    |  |  |          identityref
        |     |                    |  |  +--rw cipher-suites
        |     |                    |  |     +--rw cipher-suite*
        |     |                    |  |             identityref
        |     |                    |  +--rw keepalives!
        |     |                    |          {tls-client-keepalives}?
        |     |                    |     +--rw max-wait?       uint16
        |     |                    |     +--rw max-attempts?   uint8
        |     |                    +--rw proxy-client-identity
        |     |                       +--rw user-id?    string
        |     |                       +--rw password?   string
        |     +--rw connection-type
        |     |  +--rw (connection-type)
        |     |     +--:(persistent-connection)
        |     |     |  +--rw persistent!
        |     |     +--:(periodic-connection)
        |     |        +--rw periodic!
        |     |           +--rw period?         uint16
        |     |           +--rw anchor-time?    yang:date-and-time
        |     |           +--rw idle-timeout?   uint16
        |     +--rw reconnect-strategy
        |        +--rw start-with?     enumeration
        |        +--rw max-attempts?   uint8
        +--rw listen! {https-listen}?
           +--rw idle-timeout?   uint16
           +--rw endpoint* [name]
              +--rw name           string
              +--rw (transport)
                 +--:(https) {https-listen}?
                    +--rw https
                       +--rw tcp-server-parameters
                       |  +--rw local-address    inet:ip-address
                       |  +--rw local-port?      inet:port-number
                       |  +--rw keepalives! {keepalives-supported}?
                       |     +--rw idle-time         uint16
```

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

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

```
                        |     |          {scram-sha-256-auth}?
                        |     +--:(vapid)
                        |        +--rw vapid {vapid-auth}?
                        +--rw proxy-server! {proxy-connect}?
                           +--rw tcp-client-parameters
                           |  +--rw remote-address    inet:host
                           |  +--rw remote-port?      inet:port-number
                           |  +--rw local-address?    inet:ip-address
                           |  |       {local-binding-supported}?
                           |  +--rw local-port?       inet:port-number
                           |  |       {local-binding-supported}?
                           |  +--rw keepalives!
                           |          {keepalives-supported}?
                           |     +--rw idle-time         uint16
                           |     +--rw max-probes        uint16
                           |     +--rw probe-interval    uint16
                           +--rw tls-client-parameters
                           |  +--rw client-identity
                           |  |  +--rw (local-or-keystore)
                           |  |     +--:(local)
                           |  |     |        {local-definitions-suppo\
   \rted}?
                           |  |     |  +--rw local-definition
                           |  |     |     +--rw algorithm
                           |  |     |     |       asymmetric-key-algo\
   \rithm-t
                           |  |     |     +--rw public-key
                           |  |     |     |       binary
                           |  |     |     +--rw (private-key-type)
                           |  |     |     |  +--:(private-key)
                           |  |     |     |  |  +--rw private-key?
                           |  |     |     |  |          binary
                           |  |     |     |  +--:(hidden-private-key)
                           |  |     |     |  |  +--rw hidden-private-\
   \key?
                           |  |     |     |  |          empty
                           |  |     |     |  +--:(encrypted-private-k\
   \ey)
                           |  |     |     |     +--rw encrypted-priva\
   \te-key
                           |  |     |     |        +--rw (key-type)
                           |  |     |     |        |  +--:(symmetric-\
   \key-ref)
                           |  |     |     |        |  |  +--rw symmet\
   \ric-key-ref?    leafref
                           |  |     |     |        |  |          {key\
   \store-supported}?
                           |  |     |     |        |  +--:(asymmetric\
```

```
   \-key-ref)
                                         |  |     |     |        |     +--rw asymme\
   \tric-key-ref?   leafref
                                         |  |     |     |        |            {key\
   \store-supported}?
                                         |  |     |     |           +--rw value?
                                         |  |     |     |                  binary
                                         |  |     |     +--rw cert?
                                         |  |     |     |       end-entity-cert-cms
                                         |  |     |     +---n certificate-expiration
                                         |  |     |     |  +-- expiration-date
                                         |  |     |     |          yang:date-and-ti\
   \me
                                         |  |     |     +---x generate-certificate-\
   \signing-request
                                         |  |     |        +---w input
                                         |  |     |        |  +---w subject
                                         |  |     |        |  |       binary
                                         |  |     |        |  +---w attributes?
                                         |  |     |        |          binary
                                         |  |     |        +--ro output
                                         |  |     |           +--ro certificate-sig\
   \ning-request
                                         |  |     |                       binary
                                         |  |     +--:(keystore)
                                         |  |              {keystore-supported}?
                                         |  |        +--rw keystore-reference
                                         |  |           +--rw asymmetric-key?
                                         |  |           |       ks:asymmetric-key-r\
   \ef
                                         |  |           +--rw certificate?      lea\
   \fref
                                         |  +--rw server-authentication
                                         |  |  +--rw ca-certs?
                                         |  |  |       ts:certificates-ref
                                         |  |  |       {ts:x509-certificates}?
                                         |  |  +--rw server-certs?
                                         |  |          ts:certificates-ref
                                         |  |          {ts:x509-certificates}?
                                         |  +--rw hello-params
                                         |  |       {tls-client-hello-params-config\
   \}?
                                         |  |  +--rw tls-versions
                                         |  |  |  +--rw tls-version*   identityref
                                         |  |  +--rw cipher-suites
                                         |  |     +--rw cipher-suite*   identityref
                                         |  +--rw keepalives!
                                         |          {tls-client-keepalives}?
```

```
                             |     +--rw max-wait?       uint16
                             |     +--rw max-attempts?   uint8
                             +--rw proxy-client-identity
                                +--rw user-id?    string
                                +--rw password?   string
```

A.2. Expanded Tree Diagram for 'ietf-restconf-server'

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

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

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

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

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

```
        |        |     |     |             empty
        |        |     |     +--rw (local-or-external)
        |        |     |        +--:(local)
        |        |     |        |        {local-client-auth-supported}?
        |        |     |        |  +--rw users
        |        |     |        |     +--rw user* [name]
        |        |     |        |        +--rw name        string
        |        |     |        |        +--rw password?
        |        |     |        |                ianach:crypt-hash
        |        |     |        +--:(external)
        |        |     |                 {external-client-auth-supporte\
   d}?
        |        |     |           +--rw client-auth-defined-elsewhere?
        |        |     |                   empty
        |        |     +--rw restconf-server-parameters
        |        |        +--rw client-identification
        |        |           +--rw cert-maps
        |        |              +--rw cert-to-name* [id]
        |        |                 +--rw id             uint32
        |        |                 +--rw fingerprint
        |        |                 |       x509c2n:tls-fingerprint
        |        |                 +--rw map-type       identityref
        |        |                 +--rw name           string
        |        +--:(https) {https-listen}?
        |           +--rw https
        |              +--rw tcp-server-parameters
        |              |  +--rw local-address    inet:ip-address
        |              |  +--rw local-port?      inet:port-number
        |              |  +--rw keepalives! {keepalives-supported}?
        |              |     +--rw idle-time         uint16
        |              |     +--rw max-probes        uint16
        |              |     +--rw probe-interval    uint16
        |              +--rw tls-server-parameters
        |              |  +--rw server-identity
        |              |  |  +--rw (local-or-keystore)
        |              |  |     +--:(local)
        |              |  |     |        {local-definitions-supported}?
        |              |  |     |  +--rw local-definition
        |              |  |     |     +--rw algorithm
        |              |  |     |     |       asymmetric-key-algorithm-t
        |              |  |     |     +--rw public-key
        |              |  |     |     |       binary
        |              |  |     |     +--rw (private-key-type)
        |              |  |     |     |  +--:(private-key)
        |              |  |     |     |  |  +--rw private-key?
        |              |  |     |     |  |          binary
        |              |  |     |     |  +--:(hidden-private-key)
        |              |  |     |     |  |  +--rw hidden-private-key?
```

```
        |              |  |     |     |  |          empty
        |              |  |     |     |  +--:(encrypted-private-key)
        |              |  |     |     |     +--rw encrypted-private-key
        |              |  |     |     |        +--rw (key-type)
        |              |  |     |     |        |  +--:(symmetric-key-re\
   f)
        |              |  |     |     |        |  |  +--rw symmetric-ke\
   y-ref?    leafref
        |              |  |     |     |        |  |          {keystore-\
   supported}?
        |              |  |     |     |        |  +--:(asymmetric-key-r\
   ef)
        |              |  |     |     |        |     +--rw asymmetric-k\
   ey-ref?   leafref
        |              |  |     |     |        |             {keystore-\
   supported}?
        |              |  |     |     |        +--rw value?
        |              |  |     |     |                binary
        |              |  |     |     +--rw cert?
        |              |  |     |     |       end-entity-cert-cms
        |              |  |     |     +---n certificate-expiration
        |              |  |     |     |  +-- expiration-date
        |              |  |     |     |          yang:date-and-time
        |              |  |     |     +---x generate-certificate-signin\
   g-request
        |              |  |     |        +---w input
        |              |  |     |        |  +---w subject       binary
        |              |  |     |        |  +---w attributes?   binary
        |              |  |     |        +--ro output
        |              |  |     |           +--ro certificate-signing-r\
   equest
        |              |  |     |                   binary
        |              |  |     +--:(keystore) {keystore-supported}?
        |              |  |        +--rw keystore-reference
        |              |  |           +--rw asymmetric-key?
        |              |  |           |       ks:asymmetric-key-ref
        |              |  |           +--rw certificate?      leafref
        |              |  +--rw client-authentication!
        |              |  |  +--rw (required-or-optional)
        |              |  |  |  +--:(required)
        |              |  |  |  |  +--rw required?
        |              |  |  |  |          empty
        |              |  |  |  +--:(optional)
        |              |  |  |     +--rw optional?
        |              |  |  |             empty
        |              |  |  +--rw (local-or-external)
        |              |  |     +--:(local)
        |              |  |     |        {local-client-auth-supported}?
```

```
        |              |  |     |  +--rw ca-certs?
        |              |  |     |  |       ts:certificates-ref
        |              |  |     |  |       {ts:x509-certificates}?
        |              |  |     |  +--rw client-certs?
        |              |  |     |          ts:certificates-ref
        |              |  |     |          {ts:x509-certificates}?
        |              |  |     +--:(external)
        |              |  |              {external-client-auth-supporte\
   d}?
        |              |  |        +--rw client-auth-defined-elsewhere?
        |              |  |                empty
        |              |  +--rw hello-params
        |              |  |       {tls-server-hello-params-config}?
        |              |  |  +--rw tls-versions
        |              |  |  |  +--rw tls-version*   identityref
        |              |  |  +--rw cipher-suites
        |              |  |     +--rw cipher-suite*   identityref
        |              |  +--rw keepalives! {tls-server-keepalives}?
        |              |     +--rw max-wait?       uint16
        |              |     +--rw max-attempts?   uint8
        |              +--rw http-server-parameters
        |              |  +--rw server-name?             string
        |              |  +--rw protocol-versions
        |              |  |  +--rw protocol-version*   enumeration
        |              |  +--rw client-authentication!
        |              |     +--rw (required-or-optional)
        |              |     |  +--:(required)
        |              |     |  |  +--rw required?
        |              |     |  |          empty
        |              |     |  +--:(optional)
        |              |     |     +--rw optional?
        |              |     |             empty
        |              |     +--rw (local-or-external)
        |              |        +--:(local)
        |              |        |        {local-client-auth-supported}?
        |              |        |  +--rw users
        |              |        |     +--rw user* [name]
        |              |        |        +--rw name        string
        |              |        |        +--rw password?
        |              |        |                ianach:crypt-hash
        |              |        +--:(external)
        |              |                 {external-client-auth-supporte\
   d}?
        |              |           +--rw client-auth-defined-elsewhere?
        |              |                   empty
        |              +--rw restconf-server-parameters
        |                 +--rw client-identification
        |                    +--rw cert-maps
```

```
     |                       +--rw cert-to-name* [id]
     |                          +--rw id             uint32
     |                          +--rw fingerprint
     |                          |       x509c2n:tls-fingerprint
     |                          +--rw map-type       identityref
     |                          +--rw name           string
     +--rw call-home! {https-call-home}?
        +--rw restconf-client* [name]
           +--rw name                   string
           +--rw endpoints
           |  +--rw endpoint* [name]
           |     +--rw name           string
           |     +--rw (transport)
           |        +--:(https) {https-listen}?
           |           +--rw https
           |              +--rw tcp-client-parameters
           |              |  +--rw remote-address    inet:host
           |              |  +--rw remote-port?      inet:port-number
           |              |  +--rw local-address?    inet:ip-address
           |              |  |       {local-binding-supported}?
           |              |  +--rw local-port?       inet:port-number
           |              |  |       {local-binding-supported}?
           |              |  +--rw keepalives!
           |              |          {keepalives-supported}?
           |              |     +--rw idle-time         uint16
           |              |     +--rw max-probes        uint16
           |              |     +--rw probe-interval    uint16
           |              +--rw tls-server-parameters
           |              |  +--rw server-identity
           |              |  |  +--rw (local-or-keystore)
           |              |  |     +--:(local)
           |              |  |     |        {local-definitions-suppo\
rted}?
           |              |  |     |  +--rw local-definition
           |              |  |     |     +--rw algorithm
           |              |  |     |     |       asymmetric-key-algo\
rithm-t
           |              |  |     |     +--rw public-key
           |              |  |     |     |       binary
           |              |  |     |     +--rw (private-key-type)
           |              |  |     |     |  +--:(private-key)
           |              |  |     |     |  |  +--rw private-key?
           |              |  |     |     |  |          binary
           |              |  |     |     |  +--:(hidden-private-key)
           |              |  |     |     |  |  +--rw hidden-private-\
key?
           |              |  |     |     |  |          empty
           |              |  |     |     |  +--:(encrypted-private-k\
```

```
   ey)
              |              |  |     |     |     +--rw encrypted-priva\
   te-key
              |              |  |     |     |        +--rw (key-type)
              |              |  |     |     |        |  +--:(symmetric-\
   key-ref)
              |              |  |     |     |        |  |  +--rw symmet\
   ric-key-ref?    leafref
              |              |  |     |     |        |  |          {key\
   store-supported}?
              |              |  |     |     |        |  +--:(asymmetric\
   -key-ref)
              |              |  |     |     |        |     +--rw asymme\
   tric-key-ref?   leafref
              |              |  |     |     |        |             {key\
   store-supported}?
              |              |  |     |     |        +--rw value?
              |              |  |     |     |                binary
              |              |  |     |     +--rw cert?
              |              |  |     |     |       end-entity-cert-cms
              |              |  |     |     +---n certificate-expiration
              |              |  |     |     |  +-- expiration-date
              |              |  |     |     |          yang:date-and-ti\
   me
              |              |  |     |     +---x generate-certificate-\
   signing-request
              |              |  |     |        +---w input
              |              |  |     |        |  +---w subject
              |              |  |     |        |  |       binary
              |              |  |     |        |  +---w attributes?
              |              |  |     |        |          binary
              |              |  |     |        +--ro output
              |              |  |     |           +--ro certificate-sig\
   ning-request
              |              |  |     |                   binary
              |              |  |     +--:(keystore)
              |              |  |              {keystore-supported}?
              |              |  |        +--rw keystore-reference
              |              |  |           +--rw asymmetric-key?
              |              |  |           |       ks:asymmetric-key-r\
   ef
              |              |  |           +--rw certificate?      lea\
   fref
              |              |  +--rw client-authentication!
              |              |  |  +--rw (required-or-optional)
              |              |  |  |  +--:(required)
              |              |  |  |  |  +--rw required?
              |              |  |  |  |          empty
```

```
             |               |  |  |  +--:(optional)
             |               |  |  |     +--rw optional?
             |               |  |  |             empty
             |               |  |  +--rw (local-or-external)
             |               |  |     +--:(local)
             |               |  |     |        {local-client-auth-suppo\
   rted}?
             |               |  |     |  +--rw ca-certs?
             |               |  |     |  |       ts:certificates-ref
             |               |  |     |  |       {ts:x509-certificates}?
             |               |  |     |  +--rw client-certs?
             |               |  |     |          ts:certificates-ref
             |               |  |     |          {ts:x509-certificates}?
             |               |  |     +--:(external)
             |               |  |              {external-client-auth-su\
   pported}?
             |               |  |        +--rw client-auth-defined-else\
   where?
             |               |  |                empty
             |               |  +--rw hello-params
             |               |  |       {tls-server-hello-params-config\
   }?
             |               |  |  +--rw tls-versions
             |               |  |  |  +--rw tls-version*   identityref
             |               |  |  +--rw cipher-suites
             |               |  |     +--rw cipher-suite*   identityref
             |               |  +--rw keepalives!
             |               |          {tls-server-keepalives}?
             |               |     +--rw max-wait?       uint16
             |               |     +--rw max-attempts?   uint8
             |               +--rw http-server-parameters
             |               |  +--rw server-name?             string
             |               |  +--rw protocol-versions
             |               |  |  +--rw protocol-version*   enumeration
             |               |  +--rw client-authentication!
             |               |     +--rw (required-or-optional)
             |               |     |  +--:(required)
             |               |     |  |  +--rw required?
             |               |     |  |          empty
             |               |     |  +--:(optional)
             |               |     |     +--rw optional?
             |               |     |             empty
             |               |     +--rw (local-or-external)
             |               |        +--:(local)
             |               |        |        {local-client-auth-suppo\
   rted}?
             |               |        |  +--rw users
             |               |        |     +--rw user* [name]
```

```
             |              |        |        +--rw name        string
             |              |        |        +--rw password?
             |              |        |                ianach:crypt-hash
             |              |        +--:(external)
             |              |                 {external-client-auth-su\
   pported}?
             |              |           +--rw client-auth-defined-else\
   where?
             |              |                   empty
             |              +--rw restconf-server-parameters
             |                 +--rw client-identification
             |                    +--rw cert-maps
             |                       +--rw cert-to-name* [id]
             |                          +--rw id             uint32
             |                          +--rw fingerprint
             |                          |       x509c2n:tls-fingerprint
             |                          +--rw map-type
             |                          |       identityref
             |                          +--rw name           string
             +--rw connection-type
             |  +--rw (connection-type)
             |     +--:(persistent-connection)
             |     |  +--rw persistent!
             |     +--:(periodic-connection)
             |        +--rw periodic!
             |           +--rw period?         uint16
             |           +--rw anchor-time?    yang:date-and-time
             |           +--rw idle-timeout?   uint16
             +--rw reconnect-strategy
                +--rw start-with?     enumeration
                +--rw max-attempts?   uint8
```

Appendix B. Change Log

B.1. 00 to 01

- Renamed "keychain" to "keystore".

B.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'.

B.3. 02 to 03

- Refined use of tls-client-grouping to add a must statement indicating that the TLS client must specify a client-certificate.

- Changed restconf-client??? to be a grouping (not a container).

B.4. 03 to 04

- Added RFC 8174 to Requirements Language Section.

- Replaced refine statement in ietf-restconf-client to add a mandatory true.

- Added refine statement in ietf-restconf-server to add a must statement.

- Now there are containers and groupings, for both the client and server models.

- Now tree diagrams reference ietf-netmod-yang-tree-diagrams

- Updated examples to inline key and certificates (no longer a leafref to keystore)

B.5. 04 to 05

- Now tree diagrams reference ietf-netmod-yang-tree-diagrams

- Updated examples to inline key and certificates (no longer a leafref to keystore)

B.6. 05 to 06

- Fixed change log missing section issue.

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

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

B.7. 06 to 07

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

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

o Replaced "connection-type" choice default (persistent) with "mandatory true".

o Reduced the periodic-connection's "idle-timeout" from 5 to 2 minutes.

o Replaced reconnect-timeout with period/anchor-time combo.

B.8. 07 to 08

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

B.9. 08 to 09

o Corrected use of "mandatory true" for "address" leafs.

o Updated examples to reflect update to groupings defined in the keystore draft.

o Updated to use groupings defined in new TCP and HTTP drafts.

o Updated copyright date, boilerplate template, affiliation, and folding algorithm.

B.10. 09 to 10

o Reformatted YANG modules.

B.11. 10 to 11

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

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

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

o Moved "expanded" tree diagrams to the Appendix.

B.12. 11 to 12

o Removed the 'must' statement limiting keepalives in periodic connections.

o Updated models and examples to reflect removal of the "demux" containers in the imported models.

- o Updated the "periodic-connnection" description statements to better describe behavior when connections are not closed gracefully.

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

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

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

B.13. 12 to 13

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

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

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

B.14. 13 to 14

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

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

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

Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by first name): Alan Luchuk, Andy Bierman, Balazs Kovacs, Benoit Claise, Bert Wijnen David Lamparter,

Juergen Schoenwaelder, Ladislav Lhotka, Martin Bjorklund, Mehmet Ersue, Phil Shafer, Radek Krejci, Ramkumar Dhanapal, Sean Turner, and Tom Petch.

Author's Address

Kent Watsen
Watsen Networks

EMail: kent+ietf@watsen.net

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

Copyright Notice

Table of Contents

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.

```
   +--------------+                             +--------------+
   |  Subscriber  |                             |   Publisher  |
   |              |                             |              |
   |    Logical   |                             |     Logical  |
   |  Connection  |                             |   Connection |
   |  (a)  (b)    |                             |    (a)  (b)  |
   +--------------+                             +--------------+
       | RESTCONF POST (RPC:establish-subscription)   |
       |--------------------------------------------->|
       |                          HTTP 200 OK (ID,URI)|
       |<---------------------------------------------|
       |    |HTTP GET (URI)                                |
       |    |--------------------------------------------->|
       |    |                                   HTTP 200 OK|
       |    |<---------------------------------------------|
       |    |                           SSE (notif-message)|
       |    |<---------------------------------------------|
       | RESTCONF POST (RPC:modify-subscription)      |    |
       |--------------------------------------------->|    |
       |    |                              HTTP 200 OK|    |
       |<---------------------------------------------|    |
       |    |                    SSE (subscription-modified)|
       |    |<------------------------------------------(c)|
       |    |                           SSE (notif-message)|
       |    |<---------------------------------------------|
       | RESTCONF POST (RPC:delete-subscription)      |    |
       |--------------------------------------------->|    |
       |    |                              HTTP 200 OK|    |
       |<---------------------------------------------|    |
       |    |                                         |    |
       |    |                                         |    |
       (a) (b)                                       (a)  (b)
```

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"

- "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.

```
+------------+                     +-----------+
| Subscriber |                     | Publisher |
+------------+                     +-----------+
      |                                  |
      |establish-subscription            |
      |--------------------------------->|   (a)
      |     HTTP 200 OK, id#22, URI#1    |
      |<---------------------------------|   (b)
      |GET (URI#1)                       |
      |--------------------------------->|   (c)
      | HTTP 200 OK,notif-mesg (id#22)   |
      |<---------------------------------|
      |                                  |
      |                                  |
      |establish-subscription            |
      |--------------------------------->|
      |      HTTP 200 OK, id#23, URI#2   |
      |<---------------------------------|
      |GET (URI#2)                       |
      |--------------------------------->|
      |                                  |
      |                                  |
      |             notif-mesg (id#22)   |
      |<---------------------------------|
      | HTTP 200 OK,notif-mesg (id#23)   |
      |<---------------------------------|
      |                                  |
```

Figure 2: Multiple subscriptions over RESTCONF/HTTP

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

```
POST /restconf/operations
     /ietf-subscribed-notifications:establish-subscription

{
   "ietf-subscribed-notifications:input": {
      "stream-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.1.2. Modifying Dynamic Subscriptions

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

```
+------------+                 +-----------+
| Subscriber |                 | Publisher |
+------------+                 +-----------+
      |                              |
      |  notification message (id#23)|
      |<-----------------------------|
      |                              |
      |modify-subscription (id#23)   |
      |----------------------------->|  (d)
      |    HTTP 400 error (with hint)|
      |<-----------------------------|  (e)
      |                              |
      |modify-subscription (id#23)   |
      |----------------------------->|
      |                 HTTP 200 OK  |
      |<-----------------------------|
      |                              |
      |           notif-mesg (id#23) |
      |<-----------------------------|
      |                              |
```

Figure 7: Interaction model for successful subscription modification

If the subscription being modified in Figure 7 is a datastore subscription as per [I-D.ietf-netconf-yang-push], the modification request made in (d) may look like that shown in Figure 8. As can be seen, the modifications being attempted are the application of a new xpath filter as well as the setting of a new periodic time interval.

```
POST /restconf/operations
     /ietf-subscribed-notifications:modify-subscription

{
 "ietf-subscribed-notifications:input": {
    "id": 23,
    "ietf-yang-push:datastore-xpath-filter":
       "/example-module:foo/example-module:bar",
    "ietf-yang-push:periodic": {
       "ietf-yang-push:period": 500
    }
  }
}
```

Figure 8: Subscription modification request (c)

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

```
HTTP status code - 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

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

Network Working Group
Internet-Draft
Obsoletes: 7895 (if approved)
Intended status: Standards Track
Expires: April 20, 2019

A. Bierman
YumaWorks
M. Bjorklund
Tail-f Systems
J. Schoenwaelder
Jacobs University
K. Watsen
Juniper Networks
R. Wilton
Cisco Systems

October 17, 2018

YANG Library
draft-ietf-netconf-rfc7895bis-07

Abstract

This document describes a YANG library that provides information about the YANG modules, datastores, and datastore schemas used by a network management server. Simple caching mechanisms are provided to allow clients to minimize retrieval of this information. This version of the YANG library supports the Network Management Datastore Architecture by listing all datastores supported by a network management server and the schema that is used by each of these datastores.

This document obsoletes RFC 7895.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

There is a need for a standard mechanism to expose which YANG modules [RFC7950], datastores and datastore schemas [RFC8342] are in use by a network management server.

This document defines the YANG module "ietf-yang-library" that provides this information. This version of the YANG library is compatible with the Network Management Datastore Architecture (NMDA) [RFC8342]. The previous version of the YANG library, defined in [RFC7895], is not compatible with the NMDA since it assumes that all datastores have exactly the same schema. This is not necessarily true in the NMDA since dynamic configuration datastores may have their own datastore schema. Furthermore, the operational state datastore may support non-configurable YANG modules in addition to the YANG modules supported by conventional configuration datastores.

The old YANG library definitions have been retained (for backwards compatibility reasons) but the definitions have been marked as deprecated. For backwards compatibility, an NMDA-supporting server SHOULD populate the deprecated "/modules-state" tree in a backwards-compatible manner. The new "/yang-library" tree would be ignored by legacy clients, while providing all the data needed for NMDA-aware clients, which would themselves ignore the "/modules-state" tree. The recommended approach to populate "/modules-state" is to report the schema for YANG modules that are configurable via conventional configuration datastores and for which config false data nodes are returned via a NETCONF <get> operation, or equivalent.

The YANG library information can be different on every server and it can change at runtime or across a server reboot. If a server implements multiple network management protocols to access the server's datastores, then each such protocol may have its own conceptual instantiation of the YANG library.

If a large number of YANG modules are utilized by a server, then the YANG library contents can be relatively large. Since the YANG library contents changes very infrequently, it is important that clients be able to cache the YANG library contents and easily identify whether their cache is out of date.

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 [RFC7950]:

- module
- submodule
- data node

This document uses the phrase "implementing a module" as defined in [RFC7950] Section 5.6.5.

The following terms are defined in [RFC8342]:

- datastore
- datastore schema
- configuration
- configuration datastore
- conventional configuration
- conventional configuration datastore
- operational state
- operational state datastore
- dynamic configuration datastore
- client and server

The following terms are used within this document:

- YANG library: A collection of YANG modules, submodules, datastores, and datastore schemas used by a server.
- YANG library content identifier: A server-generated identifier of the contents of the YANG library.

Tree diagrams used in this document use the notation defined in [RFC8340].

2. Objectives

The following information is needed by a client application (for each YANG module in the library) to fully utilize the YANG data modeling language:

- name: The name of the YANG module.
- revision: If defined in the YANG module or submodule, the revision is derived from the most recent revision statement within the module or submodule.
- submodule list: The name, and if defined, revision of each submodule used by the module must be identified.
- feature list: The name of each YANG feature supported by the server, in a given datastore schema, must be identified.

- o deviation list: The name of each YANG module with deviation statements affecting a given YANG module, in a given datastore schema, must be identified.

In addition, the following information is needed by a client application for each datastore supported by a server:

- o identity: The YANG identity for the datastore.

- o schema: The schema (i.e., the set of modules) implemented by the datastore.

In order to select one out of several possible data model designs, the following criteria were used:

1. The information must be efficient for a client to consume. Since the size of the YANG library can be quite large, it should be possible for clients to cache the YANG library information.

2. A dynamic configuration datastore must be able to implement a module or feature that is not implemented in the conventional configuration datastores.

3. It must be possible to not implement a module or feature in <operational>, even if it is implemented in some other datastore. This is required for transition purposes; a server that wants to implement <operational> should not have to implement all modules at once.

4. A given module can only be implemented in one revision in all datastores. If a module is implemented in more than one datastore, the same revision is implemented in all these datastores.

5. Multiple revisions can be used for import, if import-by revision is used.

6. It must be possible to use the YANG library by schema mount [I-D.ietf-netmod-schema-mount].

3. YANG Library Data Model

The "ietf-yang-library" YANG module provides information about the modules, submodules, datastores, and datastore schemas supported by a server. All data nodes in "ietf-yang-library" are "config false", and thus only accessible in the operational state datastore.

```
+-----------+
| datastore |
+-----------+
     |
     | has a
     V
+-----------+            +--------+                +------------+
| datastore |  union of  | module |  consists of   | modules +  |
|  schema   |----------->|  set   |--------------->| submodules |
+-----------+            +--------+                +------------+
```

Figure 1

The conceptual model of the YANG library is depicted in Figure 1. Following the NMDA, every datastore has an associated datastore schema. A datastore schema is a union of module sets and every module set is a collection of modules and submodules, including the modules and submodules used for imports. Note that multiple datastores may refer to the same datastore schema. Furthermore, it is possible that individual datastore schemas share module sets. A common use case is the operational state datastore schema which is a superset of the schema used by conventional configuration datastores.

Below is the YANG Tree Diagram for the "ietf-yang-library" module, excluding the deprecated "modules-state" tree:

```
module: ietf-yang-library
  +--ro yang-library
     +--ro module-set* [name]
     |  +--ro name                  string
     |  +--ro module* [name]
     |  |  +--ro name         yang:yang-identifier
     |  |  +--ro revision?    revision-identifier
     |  |  +--ro namespace    inet:uri
     |  |  +--ro location*    inet:uri
     |  |  +--ro submodule* [name]
     |  |  |  +--ro name        yang:yang-identifier
     |  |  |  +--ro revision?   revision-identifier
     |  |  |  +--ro location*   inet:uri
     |  |  +--ro feature*     yang:yang-identifier
     |  |  +--ro deviation*   -> ../../module/name
     |  +--ro import-only-module* [name revision]
     |     +--ro name         yang:yang-identifier
     |     +--ro revision     union
     |     +--ro namespace    inet:uri
     |     +--ro location*    inet:uri
     |     +--ro submodule* [name]
     |        +--ro name        yang:yang-identifier
     |        +--ro revision?   revision-identifier
     |        +--ro location*   inet:uri
     +--ro schema* [name]
     |  +--ro name          string
     |  +--ro module-set*   -> ../../module-set/name
     +--ro datastore* [name]
     |  +--ro name      ds:datastore-ref
     |  +--ro schema    -> ../../schema/name
     +--ro content-id    string

  notifications:
    +---n yang-library-update
       +--ro content-id    -> /yang-library/content-id
```

The "/yang-library" container holds the entire YANG library. The container has the following child nodes:

- o The "/yang-library/module-set" contains entries representing module sets. The list "/yang-library/module-set/module" enumerates the modules that belong to the module set. A module is listed together with its submodules (if any), a set of features, and any deviation modules. The list "/yang-library/module-set/import-only-module" lists all modules (and their submodules) used only for imports. The assignment of a module to a module-set is at the server’s discretion. This revision of the YANG library attaches no semantics as to which module-set a module is listed in.

- The "/yang-library/schema" list contains an entry for each datastore schema supported by the server. All conventional configuration datastores use the same "schema" list entry. A dynamic configuration datastore may use a different datastore schema from the conventional configuration datastores, and hence may require a separate "schema" entry. A "schema" entry has a leaf-list of references to entries in the "module-set" list. The schema consists of the union of all modules in all referenced module sets.

- The "/yang-library/datastore" list contains one entry for each datastore supported by the server, and it identifies the datastore schema associated with a datastore via a reference to an entry in the "schema" list. Each supported conventional configuration datastore has a separate entry, pointing to the same "schema" list element.

- The "/yang-library/content-id" leaf contains the YANG library content identifier, which is an implementation-specific identifier representing the current information in the YANG library on a specific server. The value of this leaf MUST change whenever the information in the YANG library changes. There is no requirement that the same information always results in the same "content-id" value. This leaf allows a client to fetch all schema information once, cache it, and only refetch it if the value of this leaf has been changed. If the value of this leaf changes, the server also generates a "yang-library-update" notification.

Note that for a NETCONF server implementing the NETCONF extensions to support the NMDA [I-D.ietf-netconf-nmda-netconf], a change of the YANG library content identifier results in a new value for the :yang-library:1.1 capability defined in [I-D.ietf-netconf-nmda-netconf]. Thus, if such a server implements NETCONF notifications [RFC5277], and the notification "netconf-capability-change" [RFC6470], a "netconf-capability-change" notification is generated whenever the YANG library content identifier changes.

4. YANG Library YANG Module

The "ietf-yang-library" YANG module imports definitions from "ietf-yang-types" and "ietf-inet-types" defined in [RFC6991] and from "ietf-datastores" defined in [RFC8342]. While the YANG module is defined using YANG version 1.1, the YANG library supports the YANG modules written in any version of YANG.

RFC Ed.: update the date below with the date of RFC publication and remove this note.

```
<CODE BEGINS> file "ietf-yang-library@2018-10-16.yang"

module ietf-yang-library {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-yang-library";
  prefix "yanglib";

  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-datastores {
    prefix ds;
    reference "RFC 8342: Network Management Datastore Architecture.";
  }

  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>

     Author:   Juergen Schoenwaelder
               <mailto:j.schoenwaelder@jacobs-university.de>

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

     Author:   Rob Wilton
               <mailto:rwilton@cisco.com>";

  description
    "This module provides information about the YANG modules,
     datastores, and datastore schemas used by a network
```

```
        management server.

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

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

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

     // RFC Ed.: update the date below with the date of RFC publication
     // and remove this note.
     // RFC Ed.: replace XXXX with actual RFC number and remove this
     // note.
     revision 2018-10-16 {
       description
         "Added support for multiple datastores according to the
          Network Management Datastore Architecture (NMDA).";
       reference
         "RFC XXXX: YANG Library.";
     }
     revision 2016-04-09 {
       description
         "Initial revision.";
       reference
         "RFC 7895: YANG Module Library.";
     }

     /*
      * Typedefs
      */

     typedef revision-identifier {
       type string {
         pattern '\d{4}-\d{2}-\d{2}';
       }
       description
         "Represents a specific date in YYYY-MM-DD format.";
     }

     /*
      * Groupings
      */
```

```
     grouping module-identification-leafs {
       description
         "Parameters for identifying YANG modules and submodules.";

       leaf name {
         type yang:yang-identifier;
         mandatory true;
         description
           "The YANG module or submodule name.";
       }
       leaf revision {
         type revision-identifier;
         description
           "The YANG module or submodule revision date.  If no revision
            statement is present in the YANG module or submodule, this
            leaf is not instantiated.";
       }
     }

     grouping location-leaf-list {
       description
         "Common location leaf list parameter for modules and
          submodules.";

       leaf-list location {
         type inet:uri;
         description
           "Contains a URL that represents the YANG schema
            resource for this module or submodule.

            This leaf will only be present if there is a URL
            available for retrieval of the schema for this entry.";
       }
     }

     grouping module-implementation-parameters {
       description
         "Parameters for describing the implementation of a module.";

       leaf-list feature {
         type yang:yang-identifier;
         description
           "List of all YANG feature names from this module that are
            supported by the server, regardless whether they are defined
            in the module or any included submodule.";
       }
       leaf-list deviation {
         type leafref {
```

```
          path "../../module/name";
        }
        description
          "List of all YANG deviation modules used by this server to
           modify the conformance of the module associated with this
           entry.  Note that the same module can be used for deviations
           for multiple modules, so the same entry MAY appear within
           multiple 'module' entries.

           This reference MUST NOT (directly or indirectly)
           refer to the module being deviated.

           Robust clients may want to make sure that they handle a
           situation where a module deviates itself (directly or
           indirectly) gracefully.";
      }
    }

    grouping module-set-parameters {
      description
        "A set of parameters that describe a module set.";

      leaf name {
        type string;
        description
          "An arbitrary name of the module set.";
      }
      list module {
        key "name";
        description
          "An entry in this list represents a module implemented by the
           server, as per RFC 7950 section 5.6.5, with a particular set
           of supported features and deviations.";
        reference
          "RFC 7950: The YANG 1.1 Data Modeling Language.";

        uses module-identification-leafs;

        leaf namespace {
          type inet:uri;
          mandatory true;
          description
            "The XML namespace identifier for this module.";
        }

        uses location-leaf-list;

        list submodule {
```

```
          key "name";
          description
            "Each entry represents one submodule within the
             parent module.";
          uses module-identification-leafs;
          uses location-leaf-list;
        }

        uses module-implementation-parameters;
      }
      list import-only-module {
        key "name revision";
        description
          "An entry in this list indicates that the server imports
           reusable definitions from the specified revision of the
           module, but does not implement any protocol accessible
           objects from this revision.

           Multiple entries for the same module name MAY exist.  This
           can occur if multiple modules import the same module, but
           specify different revision-dates in the import statements.";

        leaf name {
          type yang:yang-identifier;
          description
            "The YANG module name.";
        }
        leaf revision {
          type union {
            type revision-identifier;
            type string {
              length 0;
            }
          }
          description
            "The YANG module revision date.
             A zero-length string is used if no revision statement
             is present in the YANG module.";
        }
        leaf namespace {
          type inet:uri;
          mandatory true;
          description
            "The XML namespace identifier for this module.";
        }

        uses location-leaf-list;
```

```
        list submodule {
          key "name";
          description
            "Each entry represents one submodule within the
             parent module.";

          uses module-identification-leafs;
          uses location-leaf-list;
        }
      }
    }

    grouping yang-library-parameters {
      description
        "The YANG library data structure is represented as a grouping
         so it can be reused in configuration or another monitoring
         data structure.";

      list module-set {
        key name;
        description
          "A set of modules that may be used by one or more schemas.

           A module set does not have to be referentially complete,
           i.e., it may define modules that contain import statements
           for other modules not included in the module set.";

        uses module-set-parameters;
      }

      list schema {
        key "name";
        description
          "A datastore schema that may be used by one or more
           datastores.

           The schema must be valid and referentially complete, i.e.,
           it must contain modules to satisfy all used import
           statements for all modules specified in the schema.";

        leaf name {
          type string;
          description
            "An arbitrary name of the schema.";
        }
        leaf-list module-set {
          type leafref {
            path "../../module-set/name";
```

```
          }
          description
            "A set of module-sets that are included in this schema.
             If a non import-only module appears in multiple module
             sets, then the module revision and the associated features
             and deviations must be identical.";
        }
      }

      list datastore {
        key "name";
        description
          "A datastore supported by this server.

           Each datastore indicates which schema it supports.

           The server MUST instantiate one entry in this list per
           specific datastore it supports.

           Each datstore entry with the same datastore schema SHOULD
           reference the same schema.";

        leaf name {
          type ds:datastore-ref;
          description
            "The identity of the datastore.";
        }
        leaf schema {
          type leafref {
            path "../../schema/name";
          }
          mandatory true;
          description
            "A reference to the schema supported by this datastore.
             All non import-only modules of the schema are implemented
             with their associated features and deviations.";
        }
      }
    }

    /*
     * Top-level container
     */

    container yang-library {
      config false;
      description
        "Container holding the entire YANG library of this server.";
```

```
    uses yang-library-parameters;

    leaf content-id {
      type string;
      mandatory true;
      description
        "A server-generated identifier of the contents of the
         'yang-library' tree.  The server MUST change the value of
         this leaf if the information represented by the
         'yang-library' tree, except 'yang-library/content-id', has
         changed.";
    }
  }

  /*
   * Notifications
   */

  notification yang-library-update {
    description
      "Generated when any YANG library information on the
       server has changed.";

    leaf content-id {
      type leafref {
        path "/yanglib:yang-library/yanglib:content-id";
      }
      mandatory true;
      description
        "Contains the YANG library content identifier for the updated
         YANG library at the time the notification is generated.";
    }
  }

  /*
   * Legacy groupings
   */

  grouping module-list {
    status deprecated;
    description
      "The module data structure is represented as a grouping
       so it can be reused in configuration or another monitoring
       data structure.";

    grouping common-leafs {
      status deprecated;
      description
```

```
          "Common parameters for YANG modules and submodules.";

        leaf name {
          type yang:yang-identifier;
          status deprecated;
          description
            "The YANG module or submodule name.";
        }
        leaf revision {
          type union {
            type revision-identifier;
            type string {
              length 0;
            }
          }
          status deprecated;
          description
            "The YANG module or submodule revision date.
             A zero-length string is used if no revision statement
             is present in the YANG module or submodule.";
        }
      }
      grouping schema-leaf {
        status deprecated;
        description
          "Common schema leaf parameter for modules and submodules.";
        leaf schema {
          type inet:uri;
          description
            "Contains a URL that represents the YANG schema
             resource for this module or submodule.

             This leaf will only be present if there is a URL
             available for retrieval of the schema for this entry.";
        }
      }

      list module {
        key "name revision";
        status deprecated;
        description
          "Each entry represents one revision of one module
           currently supported by the server.";

        uses common-leafs {
          status deprecated;
        }
        uses schema-leaf {
```

```
          status deprecated;
        }

        leaf namespace {
          type inet:uri;
          mandatory true;
          status deprecated;
          description
            "The XML namespace identifier for this module.";
        }
        leaf-list feature {
          type yang:yang-identifier;
          status deprecated;
          description
            "List of YANG feature names from this module that are
             supported by the server, regardless whether they are
             defined in the module or any included submodule.";
        }
        list deviation {
          key "name revision";
          status deprecated;
          description
            "List of YANG deviation module names and revisions
             used by this server to modify the conformance of
             the module associated with this entry.  Note that
             the same module can be used for deviations for
             multiple modules, so the same entry MAY appear
             within multiple 'module' entries.

             The deviation module MUST be present in the 'module'
             list, with the same name and revision values.
             The 'conformance-type' value will be 'implement' for
             the deviation module.";
          uses common-leafs {
            status deprecated;
          }
        }
        leaf conformance-type {
          type enumeration {
            enum implement {
              description
                "Indicates that the server implements one or more
                 protocol-accessible objects defined in the YANG module
                 identified in this entry.  This includes deviation
                 statements defined in the module.

                 For YANG version 1.1 modules, there is at most one
                 module entry with conformance type 'implement' for a
```

```
                 particular module name, since YANG 1.1 requires that
                 at most one revision of a module is implemented.

                 For YANG version 1 modules, there SHOULD NOT be more
                 than one module entry for a particular module name.";
            }
            enum import {
              description
                "Indicates that the server imports reusable definitions
                 from the specified revision of the module, but does
                 not implement any protocol accessible objects from
                 this revision.

                 Multiple module entries for the same module name MAY
                 exist. This can occur if multiple modules import the
                 same module, but specify different revision-dates in
                 the import statements.";
            }
          }
          mandatory true;
          status deprecated;
          description
            "Indicates the type of conformance the server is claiming
             for the YANG module identified by this entry.";
        }
        list submodule {
          key "name revision";
          status deprecated;
          description
            "Each entry represents one submodule within the
             parent module.";
          uses common-leafs {
            status deprecated;
          }
          uses schema-leaf {
            status deprecated;
          }
        }
      }
    }

    /*
     * Legacy operational state data nodes
     */

    container modules-state {
      config false;
      status deprecated;
```

```
      description
        "Contains YANG module monitoring information.";

      leaf module-set-id {
        type string;
        mandatory true;
        status deprecated;
        description
          "Contains a server-specific identifier representing
           the current set of modules and submodules.  The
           server MUST change the value of this leaf if the
           information represented by the 'module' list instances
           has changed.";
      }

      uses module-list {
        status deprecated;
      }
    }

    /*
     * Legacy notifications
     */

    notification yang-library-change {
      status deprecated;
      description
        "Generated when the set of modules and submodules supported
         by the server has changed.";
      leaf module-set-id {
        type leafref {
          path "/yanglib:modules-state/yanglib:module-set-id";
        }
        mandatory true;
        status deprecated;
        description
          "Contains the module-set-id value representing the
           set of modules and submodules supported at the server
           at the time the notification is generated.";
      }
    }

  }

  <CODE ENDS>
```

5. IANA Considerations

RFC 7895 previously registered one URI in the IETF XML registry [RFC3688]. This document takes over this registration entry made by RFC 7895 and changes the Registrant to the IESG according to Section 4 in [RFC3688].

URI: urn:ietf:params:xml:ns:yang:ietf-yang-library

Registrant Contact: The IESG.

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

RFC 7895 previously registered one YANG module in the "YANG Module Names" registry [RFC6020] as follows:

```
name:         ietf-yang-library
namespace:    urn:ietf:params:xml:ns:yang:ietf-yang-library
prefix:       yanglib
reference:    RFC 7895
```

This document takes over this registration entry made by RFC 7895.

6. Security Considerations

The YANG module specified in this document defines a schema for data that is accessed by network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The NETCONF access control model [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.

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:

The "/yang-library" subtree of the YANG library may help an attacker identify the server capabilities and server implementations with known bugs since the set of YANG modules supported by a server may reveal the kind of device and the manufacturer of the device.

Although some of this information may be available to all NETCONF users via the NETCONF <hello> message (or similar messages in other management protocols), this YANG module potentially exposes additional details that could be of some assistance to an attacker. Server vulnerabilities may be specific to particular modules, module revisions, module features, or even module deviations. For example, if a particular operation on a particular data node is known to cause a server to crash or significantly degrade device performance, then the module list information will help an attacker identify server implementations with such a defect, in order to launch a denial-of-service attack on the device.

7. Acknowledgments

Contributions to this material by Andy Bierman are based upon work supported by the The Space & Terrestrial Communications Directorate (S&TCD) under Contract No. W15P7T-13-C-A616. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of The Space & Terrestrial Communications Directorate (S&TCD).

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, <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>.

[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>.

[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>.

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

8.2. Informative References

[I-D.ietf-netconf-nmda-netconf]
Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "NETCONF Extensions to Support the Network Management Datastore Architecture", draft-ietf-netconf-nmda-netconf-07 (work in progress), October 2018.

[I-D.ietf-netconf-nmda-restconf]
Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "RESTCONF Extensions to Support the Network Management Datastore Architecture", draft-ietf-netconf-nmda-restconf-05 (work in progress), October 2018.

[I-D.ietf-netmod-schema-mount]
Bjorklund, M. and L. Lhotka, "YANG Schema Mount", draft-ietf-netmod-schema-mount-11 (work in progress), August 2018.

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

[RFC6470] Bierman, A., "Network Configuration Protocol (NETCONF) Base Notifications", RFC 6470, DOI 10.17487/RFC6470, February 2012, <https://www.rfc-editor.org/info/rfc6470>.

[RFC7895] Bierman, A., Bjorklund, M., and K. Watsen, "YANG Module Library", RFC 7895, DOI 10.17487/RFC7895, June 2016, <https://www.rfc-editor.org/info/rfc7895>.

[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>.

[RFC8344] Bjorklund, M., "A YANG Data Model for IP Management", RFC 8344, DOI 10.17487/RFC8344, March 2018, <https://www.rfc-editor.org/info/rfc8344>.

[RFC8345] Clemm, A., Medved, J., Varga, R., Bahadur, N., Ananthakrishnan, H., and X. Liu, "A YANG Data Model for Network Topologies", RFC 8345, DOI 10.17487/RFC8345, March 2018, <https://www.rfc-editor.org/info/rfc8345>.

[RFC8348] Bierman, A., Bjorklund, M., Dong, J., and D. Romascanu, "A YANG Data Model for Hardware Management", RFC 8348, DOI 10.17487/RFC8348, March 2018, <https://www.rfc-editor.org/info/rfc8348>.

[RFC8349] Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", RFC 8349, DOI 10.17487/RFC8349, March 2018, <https://www.rfc-editor.org/info/rfc8349>.

Appendix A. Summary of Changes from RFC 7895

This document updates [RFC7895] in the following ways:

- o Renamed document title from "YANG Module Library" to "YANG Library".

o Added a new top-level "/yang-library" container to hold the entire YANG library providing information about module sets, schemas, and datastores.

o Refactored the "/modules-state" container into a new "/yang-library/module-set" list.

o Added a new "/yang-library/schema" list and a new "/yang-library/datastore" list.

o Added a set of new groupings as replacements for the deprecated groupings.

o Added a "yang-library-update" notification as a replacement for the deprecated "yang-library-change" notification.

o Deprecated the "/modules-state" tree.

o Deprecated the "/module-list" grouping.

o Deprecated the "/yang-library-change" notification.

Appendix B. Example YANG Library Instance for a Basic Server

The following example shows the YANG Library of a basic server implementing the "ietf-interfaces" [RFC8343] and "ietf-ip" [RFC8344] modules in the <running>, <startup>, and <operational> datastores and the "ietf-hardware" [RFC8348] module in the <operational> datastore.

Newlines in leaf values are added for formatting reasons.

```
<yang-library
    xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-library"
    xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">

  <module-set>
    <name>config-modules</name>
    <module>
      <name>ietf-interfaces</name>
      <revision>2018-01-09</revision> <!-- RFC Ed. update this -->
      <namespace>
        urn:ietf:params:xml:ns:yang:ietf-interfaces
      </namespace>
    </module>
    <module>
      <name>ietf-ip</name>
      <revision>2018-01-09</revision> <!-- RFC Ed. update this -->
      <namespace>
```

```
          urn:ietf:params:xml:ns:yang:ietf-ip
        </namespace>
      </module>
      <import-only-module>
        <name>ietf-yang-types</name>
        <revision>2013-07-15</revision>
        <namespace>
          urn:ietf:params:xml:ns:yang:ietf-yang-types
        </namespace>
      </import-only-module>
      <import-only-module>
        <name>ietf-inet-types</name>
        <revision>2013-07-15</revision>
        <namespace>
          urn:ietf:params:xml:ns:yang:ietf-inet-types
        </namespace>
      </import-only-module>
    </module-set>

    <module-set>
      <name>state-modules</name>
      <module>
        <name>ietf-hardware</name>
        <revision>2018-12-18</revision> <!-- RFC Ed. update this -->
        <namespace>
          urn:ietf:params:xml:ns:yang:ietf-hardware
        </namespace>
      </module>
      <import-only-module>
        <name>ietf-inet-types</name>
        <revision>2013-07-15</revision>
        <namespace>
          urn:ietf:params:xml:ns:yang:ietf-inet-types
        </namespace>
      </import-only-module>
      <import-only-module>
        <name>ietf-yang-types</name>
        <revision>2013-07-15</revision>
        <namespace>
          urn:ietf:params:xml:ns:yang:ietf-yang-types
        </namespace>
      </import-only-module>
      <import-only-module>
        <name>iana-hardware</name>
        <revision>2017-12-18</revision> <!-- RFC Ed. update this -->
        <namespace>
          urn:ietf:params:xml:ns:yang:iana-hardware
        </namespace>
```

```
      </import-only-module>
    </module-set>

    <schema>
      <name>config-schema</name>
      <module-set>config-modules</module-set>
    </schema>
    <schema>
      <name>state-schema</name>
      <module-set>config-modules</module-set>
      <module-set>state-modules</module-set>
    </schema>

    <datastore>
      <name>ds:startup</name>
      <schema>config-schema</schema>
    </datastore>
    <datastore>
      <name>ds:running</name>
      <schema>config-schema</schema>
    </datastore>
    <datastore>
      <name>ds:operational</name>
      <schema>state-schema</schema>
    </datastore>

    <content-id>75a43df9bd56b92aacc156a2958fbe12312fb285</content-id>
  </yang-library>
```


Appendix C. Example YANG Library Instance for an Advanced Server

The following example extends the preceding Basic Server YANG Library example, by using modules from [RFC8345] and [RFC8349], to illustrate a slightly more advanced server that:

o Has a module with features only enabled in <operational>; the "ietf-routing module" is supported in <running>, <startup>, and <operational>, but the "multiple-ribs" and "router-id" features are only enabled in <operational>. Hence the "router-id" leaf may be read but not configured.

o Supports a dynamic configuration datastore "example-ds-ephemeral", with only the "ietf-network" and "ietf-network-topology" modules configurable via a notional dynamic configuration protocol.

o Shows an example of datastore specific deviations. The module "example-vendor-hardware-deviations" is included in the schema for

<operational> to remove data nodes that cannot be supported by the server.

o Shows how module-sets can be used to organize related modules together.

```
<yang-library
    xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-library"
    xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores"
    xmlns:ex-ds-eph="urn:example:ds-ephemeral">

  <module-set>
    <name>config-state-modules</name>
    <module>
      <name>ietf-interfaces</name>
      <revision>2018-01-09</revision> <!-- RFC Ed. update this -->
      <namespace>
        urn:ietf:params:xml:ns:yang:ietf-interfaces
      </namespace>
    </module>
    <module>
      <name>ietf-ip</name>
      <revision>2018-01-09</revision> <!-- RFC Ed. update this -->
      <namespace>
        urn:ietf:params:xml:ns:yang:ietf-ip
      </namespace>
    </module>
    <module>
      <name>ietf-routing</name>
      <revision>2018-01-25</revision> <!-- RFC Ed. update this -->
      <namespace>
        urn:ietf:params:xml:ns:yang:ietf-routing
      </namespace>
    </module>
    <import-only-module>
      <name>ietf-yang-types</name>
      <revision>2013-07-15</revision>
      <namespace>
        urn:ietf:params:xml:ns:yang:ietf-yang-types
      </namespace>
    </import-only-module>
    <import-only-module>
      <name>ietf-inet-types</name>
      <revision>2013-07-15</revision>
      <namespace>
        urn:ietf:params:xml:ns:yang:ietf-inet-types
      </namespace>
    </import-only-module>
```

```
  </module-set>

  <module-set>
    <name>config-only-modules</name>
    <module>
      <name>ietf-routing</name>
      <revision>2018-01-25</revision> <!-- RFC Ed. update this -->
      <namespace>
        urn:ietf:params:xml:ns:yang:ietf-routing
      </namespace>
    </module>
  </module-set>

  <module-set>
    <name>dynamic-config-state-modules</name>
    <module>
      <name>ietf-network</name>
      <revision>2017-12-18</revision> <!-- RFC Ed. update this -->
      <namespace>
        urn:ietf:params:xml:ns:yang:ietf-network
      </namespace>
    </module>
    <module>
      <name>ietf-network-topology</name>
      <revision>2017-12-18</revision> <!-- RFC Ed. update this -->
      <namespace>
        urn:ietf:params:xml:ns:yang:ietf-network-topology
      </namespace>
    </module>
    <import-only-module>
      <name>ietf-inet-types</name>
      <revision>2013-07-15</revision>
      <namespace>
        urn:ietf:params:xml:ns:yang:ietf-inet-types
      </namespace>
    </import-only-module>
  </module-set>

  <module-set>
    <name>state-only-modules</name>
    <module>
      <name>ietf-hardware</name>
      <revision>2018-12-18</revision> <!-- RFC Ed. update this -->
      <namespace>
        urn:ietf:params:xml:ns:yang:ietf-hardware
      </namespace>
      <deviation>example-vendor-hardware-deviations</deviation>
    </module>
```

```
      <module>
        <name>ietf-routing</name>
        <revision>2018-01-25</revision> <!-- RFC Ed. update this -->
        <namespace>
          urn:ietf:params:xml:ns:yang:ietf-routing
        </namespace>
        <feature>multiple-ribs</feature>
        <feature>router-id</feature>
      </module>
      <module>
        <name>example-vendor-hardware-deviations</name>
        <revision>2018-01-31</revision>
        <namespace>
          urn:example:example-vendor-hardware-deviations
        </namespace>
      </module>
      <import-only-module>
        <name>ietf-inet-types</name>
        <revision>2013-07-15</revision>
        <namespace>
          urn:ietf:params:xml:ns:yang:ietf-inet-types
        </namespace>
      </import-only-module>
      <import-only-module>
        <name>ietf-yang-types</name>
        <revision>2013-07-15</revision>
        <namespace>
          urn:ietf:params:xml:ns:yang:ietf-yang-types
        </namespace>
      </import-only-module>
      <import-only-module>
        <name>iana-hardware</name>
        <revision>2017-12-18</revision> <!-- RFC Ed. update this -->
        <namespace>
          urn:ietf:params:xml:ns:yang:iana-hardware
        </namespace>
      </import-only-module>
    </module-set>

    <schema>
      <name>config-schema</name>
      <module-set>config-state-modules</module-set>
      <module-set>config-only-modules</module-set>
    </schema>
    <schema>
      <name>dynamic-config-schema</name>
      <module-set>dynamic-config-state-modules</module-set>
    </schema>
```

```
    <schema>
      <name>state-schema</name>
      <module-set>config-state-modules</module-set>
      <module-set>dynamic-config-state-modules</module-set>
      <module-set>state-only-modules</module-set>
    </schema>

    <datastore>
      <name>ds:startup</name>
      <schema>config-schema</schema>
    </datastore>
    <datastore>
      <name>ds:running</name>
      <schema>config-schema</schema>
    </datastore>
    <datastore>
      <name>ex-ds-eph:ds-ephemeral</name>
      <schema>dynamic-config-schema</schema>
    </datastore>
    <datastore>
      <name>ds:operational</name>
      <schema>state-schema</schema>
    </datastore>

    <content-id>14782ab9bd56b92aacc156a2958fbe12312fb285</content-id>
  </yang-library>
```


Authors' Addresses

Andy Bierman
YumaWorks

Email: andy@yumaworks.com

Martin Bjorklund
Tail-f Systems

Email: mbj@tail-f.com

Juergen Schoenwaelder
Jacobs University

Email: j.schoenwaelder@jacobs-university.de

Kent Watsen
Juniper Networks

Email: kwatsen@juniper.net

Robert Wilton
Cisco Systems

Email: rwilton@cisco.com

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 9, 2019

K. Watsen
Watsen Networks
G. Wu
Cisco Systems
L. Xia
Huawei

June 7, 2019

YANG Groupings for SSH Clients and SSH Servers
draft-ietf-netconf-ssh-client-server-14

Abstract

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

Editorial Note (To be removed by RFC Editor)

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

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

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

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

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

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

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

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

o Appendix A. Change Log

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

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

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

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

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

2. Terminology

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

3. The SSH Client Model

3.1. Tree Diagram

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

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

module: ietf-ssh-client

  grouping ssh-client-grouping
    +-- client-identity
    |  +-- username?            string
    |  +-- (auth-type)
    |     +--:(password)
    |     |  +-- password?      string
    |     +--:(public-key)
    |     |  +-- public-key
    |     |     +---u ks:local-or-keystore-asymmetric-key-grouping
    |     +--:(certificate)
    |        +-- certificate {sshcmn:ssh-x509-certs}?
    |           +---u ks:local-or-keystore-end-entity-cert-with-key-\
grouping
    +-- server-authentication
    |  +-- ssh-host-keys?   ts:host-keys-ref {ts:ssh-host-keys}?
    |  +-- ca-certs?        ts:certificates-ref
    |  |       {sshcmn:ssh-x509-certs,ts:x509-certificates}?
    |  +-- server-certs?    ts:certificates-ref
    |          {sshcmn:ssh-x509-certs,ts:x509-certificates}?
    +-- transport-params {ssh-client-transport-params-config}?
    |  +---u sshcmn:transport-params-grouping
    +-- keepalives! {ssh-client-keepalives}?
       +-- max-wait?       uint16
       +-- max-attempts?   uint8
```

3.2. Example Usage

This section presents two examples showing the ssh-client-grouping populated with some data. These examples are effectively the same except the first configures the client identity using a local key while the second uses a key configured in a keystore. Both examples are consistent with the examples presented in Section 2 of [I-D.ietf-netconf-trust-anchors] and Section 3.2 of [I-D.ietf-netconf-keystore].

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

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

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

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

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

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

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

   </ssh-client>
```

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

```
<ssh-client
  xmlns="urn:ietf:params:xml:ns:yang:ietf-ssh-client"
  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>
      <keystore-reference>ex-rsa-key</keystore-reference>
    </public-key>
  </client-identity>

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

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

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

</ssh-client>
```

3.3. YANG Module

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

```
<CODE BEGINS> file "ietf-ssh-client@2019-06-07.yang"
module ietf-ssh-client {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-ssh-client";
  prefix sshc;

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

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

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

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

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

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

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

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

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

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

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

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

     // Features

     feature ssh-client-transport-params-config {
       description
         "SSH transport layer parameters are configurable on an SSH
          client.";
     }

     feature ssh-client-keepalives {
       description
         "Per socket SSH keepalive parameters are configurable for
          SSH clients on the server implementing this feature.";
     }

     // Groupings

     grouping ssh-client-grouping {
       description
         "A reusable grouping for configuring a SSH client without
          any consideration for how an underlying TCP session is
```

```
          established.

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

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

```
             reference
               "RFC ZZZZ: YANG Data Model for a Centralized
                          Keystore Mechanism";
           }
         }
       } // container client-identity

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

```
       } // container server-authentication

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

       container keepalives {
         nacm:default-deny-write;
         if-feature "ssh-client-keepalives";
         presence "Indicates that keepalives are enabled.";
         description
           "Configures the keep-alive policy, to proactively test
            the aliveness of the SSH server.  An unresponsive TLS
            server is dropped after approximately max-wait *
            max-attempts seconds.";
         leaf max-wait {
           type uint16 {
             range "1..max";
           }
           units "seconds";
           default "30";
           description
             "Sets the amount of time in seconds after which if
              no data has been received from the SSH server, a
              TLS-level message will be sent to test the
              aliveness of the SSH server.";
         }
         leaf max-attempts {
           type uint8;
           default "3";
           description
             "Sets the maximum number of sequential keep-alive
              messages that can fail to obtain a response from
              the SSH server before assuming the SSH server is
              no longer alive.";
         }
       } // container keepalives
     } // grouping ssh-client-grouping
   }
   <CODE ENDS>
```

4. The SSH Server Model

4.1. Tree Diagram

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

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

module: ietf-ssh-server

  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
    |  +-- supported-authentication-methods
    |  |  +-- publickey?   empty
    |  |  +-- passsword?   empty
    |  |  +-- hostbased?   empty
    |  |  +-- none?        empty
    |  |  +-- other*       string
    |  +-- (local-or-external)
    |     +--:(local) {local-client-auth-supported}?
    |     |  +-- users
    |     |     +-- user* [name]
    |     |        +-- name?             string
    |     |        +-- password?         ianach:crypt-hash
    |     |        +-- authorized-key* [name]
    |     |           +-- name?        string
    |     |           +-- algorithm    string
    |     |           +-- key-data     binary
    |     +--:(external) {external-client-auth-supported}?
    |        +-- client-auth-defined-elsewhere?   empty
    +-- transport-params {ssh-server-transport-params-config}?
    |  +---u sshcmn:transport-params-grouping
    +-- keepalives! {ssh-server-keepalives}?
       +-- max-wait?       uint16
       +-- max-attempts?   uint8
```

4.2. Example Usage

This section presents two examples showing the ssh-server-grouping populated with some data. These examples are effectively the same except the first configures the server identity using a local key while the second uses a key configured in a keystore. Both examples are consistent with the examples presented in Section 2 of [I-D.ietf-netconf-trust-anchors] and Section 3.2 of [I-D.ietf-netconf-keystore].

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

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

<ssh-server
  xmlns="urn:ietf:params:xml:ns:yang:ietf-ssh-server"
  xmlns:algs="urn:ietf:params:xml:ns:yang:ietf-ssh-common">

  <!-- which host-keys will this SSH server present -->
  <server-identity>
    <host-key>
      <name>deployment-specific-certificate</name>
      <public-key>
        <local-definition>
          <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-cryp\
to-types">ct:rsa2048</algorithm>
          <private-key>base64encodedvalue==</private-key>
          <public-key>base64encodedvalue==</public-key>
        </local-definition>
      </public-key>
    </host-key>
  </server-identity>

  <!-- which client credentials will this SSH server trust -->
  <client-authentication>
    <supported-authentication-methods>
      <publickey/>
    </supported-authentication-methods>
    <!--<local-definition>-->
      <users>
        <user>
          <name>mary</name>
        </user>
      </users>
      <!--</local-definition>-->
    <!--
    <ca-certs>explicitly-trusted-client-ca-certs</ca-certs>
```

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

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

    </ssh-server>
```

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

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

      <!-- which host-keys will this SSH server present -->
      <server-identity>
        <host-key>
          <name>deployment-specific-certificate</name>
          <public-key>
            <keystore-reference>ex-rsa-key</keystore-reference>
          </public-key>
        </host-key>
      </server-identity>

      <!-- which client credentials will this SSH server trust -->
      <client-authentication>
```

```
        <supported-authentication-methods>
          <publickey/>
        </supported-authentication-methods>
        <!--<local-definition>-->
          <users>
            <user>
              <name>mary</name>
            </user>
          </users>
          <!--</local-definition>-->
        <!--
        <ca-certs>explicitly-trusted-client-ca-certs</ca-certs>
        <client-certs>explicitly-trusted-client-certs</client-certs>
        -->
      </client-authentication>

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

    </ssh-server>
```

4.3. YANG Module

This YANG module has normative references to [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore] and informative references to [RFC4253] and [RFC7317].

<CODE BEGINS> file "ietf-ssh-server@2019-06-07.yang"

```
   module ietf-ssh-server {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-ssh-server";
     prefix sshs;

     import ietf-ssh-common {
       prefix sshcmn;
       revision-date 2019-06-07; // stable grouping definitions
       reference
         "RFC XXXX: YANG Groupings for SSH Clients and SSH Servers";
     }
   /*
     import ietf-truststore {
       prefix ta;
       reference
         "RFC YYYY: A YANG Data Model for a Truststore";
     }
   */
     import ietf-keystore {
       prefix ks;
       reference
         "RFC ZZZZ: A YANG Data Model for a Keystore";
     }

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

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

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

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

     description
       "This module defines reusable groupings for SSH servers that
        can be used as a basis for specific SSH server instances.
```

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

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

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

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

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

     // Features

     feature ssh-server-transport-params-config {
       description
         "SSH transport layer parameters are configurable on an SSH
          server.";
     }

     feature ssh-server-keepalives {
       description
         "Per socket SSH keepalive parameters are configurable for
          SSH servers on the server implementing this feature.";
     }

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

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

     // Groupings

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

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

       container server-identity {
         nacm:default-deny-write;
         description
           "The list of host-keys the SSH server will present when
            establishing a SSH connection.";
         list host-key {
           key "name";
           min-elements 1;
           ordered-by user;
           description
             "An ordered list of host keys the SSH server will use to
              construct its ordered list of algorithms, when sending
              its SSH_MSG_KEXINIT message, as defined in Section 7.1
              of RFC 4253.";
           reference
             "RFC 4253: The Secure Shell (SSH) Transport Layer
                        Protocol";
           leaf name {
             type string;
             description
               "An arbitrary name for this host-key";
           }
           choice host-key-type {
             mandatory true;
             description
```

```
               "The type of host key being specified";
             container public-key {
               uses ks:local-or-keystore-asymmetric-key-grouping;
               description
                 "A locally-defined or referenced asymmetric key pair
                  to be used for the SSH server's host key.";
               reference
                 "RFC ZZZZ: YANG Data Model for a Centralized
                            Keystore Mechanism";
             }
             container certificate {
               if-feature "sshcmn:ssh-x509-certs";
               uses
               ks:local-or-keystore-end-entity-cert-with-key-grouping;
               description
                 "A locally-defined or referenced end-entity
                  certificate to be used for the SSH server's
                  host key.";
               reference
                 "RFC ZZZZ: YANG Data Model for a Centralized
                            Keystore Mechanism";
             }
           }
         }
       } // container server-identity

       container client-authentication {
         nacm:default-deny-write;
         description
           "Specifies if SSH client authentication is required or
            optional, and specifies if the SSH client authentication
            credentials are configured locally or externally.";
         container supported-authentication-methods {
           description
             "Indicates which authentication methods the server
              supports.";
           leaf publickey {
             type empty;
             description
               "Indicates that the 'publickey' method is supported.
                Note that RFC 6187 X.509v3 Certificates for SSH uses
                the 'publickey' method name.";
             reference
               "RFC 4252: The Secure Shell (SSH) Authentication
                          Protocol.
                RFC 6187: X.509v3 Certificates for Secure Shell
                          Authentication.";
           }
```

```
           leaf passsword {
             type empty;
             description
               "Indicates that the 'password' method is supported.";
             reference
               "RFC 4252: The Secure Shell (SSH) Authentication
                          Protocol.";
           }
           leaf hostbased {
             type empty;
             description
               "Indicates that the 'hostbased' method is supported.";
             reference
               "RFC 4252: The Secure Shell (SSH) Authentication
                          Protocol.";
           }
           leaf none {
             type empty;
             description
               "Indicates that the 'none' method is supported.";
             reference
               "RFC 4252: The Secure Shell (SSH) Authentication
                          Protocol.";
           }
           leaf-list other {
             type string;
             description
               "Indicates a supported method name not defined by
                RFC 4253.";
             reference
               "RFC 4252: The Secure Shell (SSH) Authentication
                          Protocol.";
           }
         }
         choice local-or-external {
           mandatory true;
           description
             "Indicates if the client credentials are configured
              locally or externally.";
           case local {
             if-feature "local-client-auth-supported";
             description
               "Client credentials are configured locally.";
             container users {
               description
                 "A list of locally configured users.";
               list user {
                 key name;
```

```
                 description
                   "The list of local users configured on this device.";

                 leaf name {
                   type string;
                   description
                    "The user name string identifying this entry.";
                 }
                 leaf password {
                   type ianach:crypt-hash;
                   description
                     "The password for this entry.";
                 }
                 list authorized-key {
                   key name;
                   description
                     "A list of public SSH keys for this user.  These
                      keys are allowed for SSH authentication, as
                      described in RFC 4253.";
                   reference
                     "RFC 4253: The Secure Shell (SSH) Transport Layer
                                Protocol";
                   leaf name {
                     type string;
                     description
                       "An arbitrary name for the SSH key.";
                   }
                   leaf algorithm {
                     type string;
                     mandatory true;
                     description
                       "The public key algorithm name for this SSH key.

                        Valid values are the values in the IANA 'Secure
                        Shell (SSH) Protocol Parameters' registry,
                        Public Key Algorithm Names.";
                     reference
                       "IANA 'Secure Shell (SSH) Protocol Parameters'
                        registry, Public Key Algorithm Names";
                   }
                   leaf key-data {
                     type binary;
                     mandatory true;
                     description
                       "The binary public key data for this SSH key, as
                        specified by RFC 4253, Section 6.6, i.e.:

                          string    certificate or public key format
```

```
                                 identifier
                       byte[n]   key/certificate data.";
                  reference
                    "RFC 4253: The Secure Shell (SSH) Transport Layer
                               Protocol";
                }
              }
            } // list user
/*
            if-feature "sshcmn:ssh-x509-certs";
            description
              "A reference to a list of certificate authority
               (CA) certificates and a reference to a list of
               client certificates.";
            leaf ca-certs {
              if-feature "ts:x509-certificates";
              type ts:certificates-ref;  // local or remote
              description
                "A reference to a list of certificate authority (CA)
                 certificates used by the SSH server to authenticate
                 SSH client certificates.  A client certificate is
                 authenticated if it has a valid chain of trust to
                 a configured CA certificate.";
              reference
                "RFC YYYY: YANG Data Model for Global Trust Anchors";
            }
            leaf client-certs {
              if-feature "ts:x509-certificates";
              type ts:certificates-ref;  // local or remote
              description
                "A reference to a list of client certificates
                 used by the SSH server to authenticate SSH
                 client certificates.  A clients certificate
                 is authenticated if it is an exact match to
                 a configured client certificate.";
              reference
                "RFC YYYY: YANG Data Model for Global Trust Anchors";
            }
*/
          } // container users
        } // case local
        case external {
          if-feature "external-client-auth-supported";
          description
            "Client credentials are configured externally, such
             as via RADIUS, RFC 7317, or another mechanism.";
          leaf client-auth-defined-elsewhere {
            type empty;
```

```
              description
                "Indicates that client credentials are configured
                 elsewhere.";
            }
          }
        } // choice local-or-external
      } // container client-authentication

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

      container keepalives {
        nacm:default-deny-write;
        if-feature "ssh-server-keepalives";
        presence "Indicates that keepalives are enabled.";
        description
          "Configures the keep-alive policy, to proactively test
           the aliveness of the SSL client.  An unresponsive SSL
           client is dropped after approximately max-wait *
           max-attempts seconds.";
        leaf max-wait {
          type uint16 {
            range "1..max";
          }
          units "seconds";
          default "30";
          description
            "Sets the amount of time in seconds after which
             if no data has been received from the SSL client,
             a SSL-level message will be sent to test the
             aliveness of the SSL client.";
        }
        leaf max-attempts {
          type uint8;
          default "3";
          description
            "Sets the maximum number of sequential keep-alive
             messages that can fail to obtain a response from
             the SSL client before assuming the SSL client is
             no longer alive.";
        }
      } // container keepalives
    } // grouping server-identity-grouping
```

```
   }
   <CODE ENDS>
```

5. The SSH Common Model

The SSH common model presented in this section contains identities and groupings common to both SSH clients and SSH servers. The transport-params-grouping can be used to configure the list of SSH transport algorithms permitted by the SSH client or SSH server. The lists of algorithms are ordered such that, if multiple algorithms are permitted by the client, the algorithm that appears first in its list that is also permitted by the server is used for the SSH transport layer connection. The ability to restrict the algorithms allowed is provided in this grouping for SSH clients and SSH servers that are capable of doing so and may serve to make SSH clients and SSH servers compliant with security policies.

[I-D.ietf-netconf-crypto-types] defines six categories of cryptographic algorithms (hash-algorithm, symmetric-key-encryption-algorithm, mac-algorithm, asymmetric-key-encryption-algorithm, signature-algorithm, key-negotiation-algorithm) and lists several widely accepted algorithms for each of them. The SSH client and server models use one or more of these algorithms. The SSH common model includes four parameters for configuring its permitted SSH algorithms, which are: host-key-alg, key-exchange-alg, encryption-alg and mac-alg. The following tables are provided, in part, to define the subset of algorithms defined in the crypto-types model used by SSH and, in part, to ensure compatibility of configured SSH cryptographic parameters for configuring its permitted SSH algorithms ("sshcmn" representing SSH common model, and "ct" representing crypto-types model which the SSH client/server model is based on):

sshcmn:host-key-alg	ct:signature-algorithm
dsa-sha1	dsa-sha1
rsa-pkcs1-sha1	rsa-pkcs1-sha1
rsa-pkcs1-sha256	rsa-pkcs1-sha256
rsa-pkcs1-sha512	rsa-pkcs1-sha512
ecdsa-secp256r1-sha256	ecdsa-secp256r1-sha256
ecdsa-secp384r1-sha384	ecdsa-secp384r1-sha384
ecdsa-secp521r1-sha512	ecdsa-secp521r1-sha512
x509v3-rsa-pkcs1-sha1	x509v3-rsa-pkcs1-sha1
x509v3-rsa2048-pkcs1-sha256	x509v3-rsa2048-pkcs1-sha1
x509v3-ecdsa-secp256r1-sha256	x509v3-ecdsa-secp256r1-sha256
x509v3-ecdsa-secp384r1-sha384	x509v3-ecdsa-secp384r1-sha384
x509v3-ecdsa-secp521r1-sha512	x509v3-ecdsa-secp521r1-sha512

Table 1 The SSH Host-key-alg Compatibility Matrix

sshcmn:key-exchange-alg	ct:key-negotiation-algorithm
diffie-hellman-group14-sha1	diffie-hellman-group14-sha1
diffie-hellman-group14-sha256	diffie-hellman-group14-sha256
diffie-hellman-group15-sha512	diffie-hellman-group15-sha512
diffie-hellman-group16-sha512	diffie-hellman-group16-sha512
diffie-hellman-group17-sha512	diffie-hellman-group17-sha512
diffie-hellman-group18-sha512	diffie-hellman-group18-sha512
ecdh-sha2-secp256r1	ecdh-sha2-secp256r1
ecdh-sha2-secp384r1	ecdh-sha2-secp384r1

Table 2 The SSH Key-exchange-alg Compatibility Matrix

sshcmn:encryption-alg	ct:symmetric-key-encryption-algorithm
aes-128-cbc	aes-128-cbc
aes-192-cbc	aes-192-cbc
aes-256-cbc	aes-256-cbc
aes-128-ctr	aes-128-ctr
aes-192-ctr	aes-192-ctr
aes-256-ctr	aes-256-ctr

Table 3 The SSH Encryption-alg Compatibility Matrix

sshcmn:mac-alg	ct:mac-algorithm
hmac-sha1	hmac-sha1
hmac-sha1-96	hmac-sha1-96
hmac-sha2-256	hmac-sha2-256
hmac-sha2-512	hmac-sha2-512

Table 4 The SSH Mac-alg Compatibility Matrix

As is seen in the tables above, the names of the "sshcmn" algorithms are all identical to the names of algorithms defined in [I-D.ietf-netconf-crypto-types]. While appearing to be redundant, it is important to realize that not all the algorithms defined in [I-D.ietf-netconf-crypto-types] are supported by SSH. That is, the algorithms supported by SSH are a subset of the algorithms defined in [I-D.ietf-netconf-crypto-types]. The algorithms used by SSH are redefined in this document in order to constrain the algorithms that may be selected to just the ones used by SSH.

Features are defined for algorithms that are OPTIONAL or are not widely supported by popular implementations. Note that the list of algorithms is not exhaustive. As well, some algorithms that are REQUIRED by [RFC4253] are missing, notably "ssh-dss" and "diffie-hellman-group1-sha1" due to their weak security and there being alternatives that are widely supported.

5.1. Tree Diagram

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

```
module: ietf-ssh-common

  grouping transport-params-grouping
    +-- host-key
    |  +-- host-key-alg*   identityref
    +-- key-exchange
    |  +-- key-exchange-alg*   identityref
    +-- encryption
    |  +-- encryption-alg*   identityref
    +-- mac
       +-- mac-alg*   identityref
```

5.2. Example Usage

This following example illustrates how the transport-params-grouping appears when populated with some data.


```
<transport-params
  xmlns="urn:ietf:params:xml:ns:yang:ietf-ssh-common"
  xmlns:algs="urn:ietf:params:xml:ns:yang:ietf-ssh-common">
  <host-key>
    <host-key-alg>algs:x509v3-rsa2048-sha256</host-key-alg>
    <host-key-alg>algs:ssh-rsa</host-key-alg>
  </host-key>
  <key-exchange>
    <key-exchange-alg>
      algs:diffie-hellman-group-exchange-sha256
    </key-exchange-alg>
  </key-exchange>
  <encryption>
    <encryption-alg>algs:aes256-ctr</encryption-alg>
    <encryption-alg>algs:aes192-ctr</encryption-alg>
    <encryption-alg>algs:aes128-ctr</encryption-alg>
    <encryption-alg>algs:aes256-cbc</encryption-alg>
    <encryption-alg>algs:aes192-cbc</encryption-alg>
    <encryption-alg>algs:aes128-cbc</encryption-alg>
  </encryption>
  <mac>
    <mac-alg>algs:hmac-sha2-256</mac-alg>
    <mac-alg>algs:hmac-sha2-512</mac-alg>
  </mac>
</transport-params>
```


5.3. YANG Module

This YANG module has normative references to [RFC4253], [RFC4344], [RFC4419], [RFC5656], [RFC6187], and [RFC6668].

```
<CODE BEGINS> file "ietf-ssh-common@2019-06-07.yang"
module ietf-ssh-common {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-ssh-common";
  prefix sshcmn;

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

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

```
        Author:   Kent Watsen <mailto:kent+ietf@watsen.net>
        Author:   Gary Wu <mailto:garywu@cisco.com>";

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

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

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

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

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

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

     // Features

     feature ssh-ecc {
       description
         "Elliptic Curve Cryptography is supported for SSH.";
       reference
         "RFC 5656: Elliptic Curve Algorithm Integration in the
                    Secure Shell Transport Layer";
     }

     feature ssh-x509-certs {
       description
         "X.509v3 certificates are supported for SSH per RFC 6187.";
```

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

     feature ssh-dh-group-exchange {
       description
         "Diffie-Hellman Group Exchange is supported for SSH.";
       reference
         "RFC 4419: Diffie-Hellman Group Exchange for the
                    Secure Shell (SSH) Transport Layer Protocol";
     }

     feature ssh-ctr {
       description
         "SDCTR encryption mode is supported for SSH.";
       reference
         "RFC 4344: The Secure Shell (SSH) Transport Layer
                    Encryption Modes";
     }

     feature ssh-sha2 {
       description
         "The SHA2 family of cryptographic hash functions is
          supported for SSH.";
       reference
         "FIPS PUB 180-4: Secure Hash Standard (SHS)";
     }

     // Identities

     identity public-key-alg-base {
       description
         "Base identity used to identify public key algorithms.";
     }

     identity ssh-dss {
       base public-key-alg-base;
       description
         "Digital Signature Algorithm using SHA-1 as the
          hashing algorithm.";
       reference
         "RFC 4253:
            The Secure Shell (SSH) Transport Layer Protocol";
     }

     identity ssh-rsa {
       base public-key-alg-base;
```

```
       description
         "RSASSA-PKCS1-v1_5 signature scheme using SHA-1 as the
          hashing algorithm.";
       reference
         "RFC 4253:
            The Secure Shell (SSH) Transport Layer Protocol";
     }

     identity ecdsa-sha2-nistp256 {
       base public-key-alg-base;
       if-feature "ssh-ecc and ssh-sha2";
       description
         "Elliptic Curve Digital Signature Algorithm (ECDSA) using the
          nistp256 curve and the SHA2 family of hashing algorithms.";
       reference
         "RFC 5656: Elliptic Curve Algorithm Integration in the
                    Secure Shell Transport Layer";
     }

     identity ecdsa-sha2-nistp384 {
       base public-key-alg-base;
       if-feature "ssh-ecc and ssh-sha2";
       description
         "Elliptic Curve Digital Signature Algorithm (ECDSA) using the
          nistp384 curve and the SHA2 family of hashing algorithms.";
       reference
         "RFC 5656: Elliptic Curve Algorithm Integration in the
                    Secure Shell Transport Layer";
     }

     identity ecdsa-sha2-nistp521 {
       base public-key-alg-base;
       if-feature "ssh-ecc and ssh-sha2";
       description
         "Elliptic Curve Digital Signature Algorithm (ECDSA) using the
          nistp521 curve and the SHA2 family of hashing algorithms.";
       reference
         "RFC 5656: Elliptic Curve Algorithm Integration in the
                    Secure Shell Transport Layer";
     }

     identity x509v3-ssh-rsa {
       base public-key-alg-base;
       if-feature "ssh-x509-certs";
       description
         "RSASSA-PKCS1-v1_5 signature scheme using a public key stored
          in an X.509v3 certificate and using SHA-1 as the hashing
          algorithm.";
```

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

     identity x509v3-rsa2048-sha256 {
       base public-key-alg-base;
       if-feature "ssh-x509-certs and ssh-sha2";
       description
         "RSASSA-PKCS1-v1_5 signature scheme using a public key stored
          in an X.509v3 certificate and using SHA-256 as the hashing
          algorithm.  RSA keys conveyed using this format MUST have a
          modulus of at least 2048 bits.";
       reference
         "RFC 6187: X.509v3 Certificates for Secure Shell
                    Authentication";
     }

     identity x509v3-ecdsa-sha2-nistp256 {
       base public-key-alg-base;
       if-feature "ssh-ecc and ssh-x509-certs and ssh-sha2";
       description
         "Elliptic Curve Digital Signature Algorithm (ECDSA)
          using the nistp256 curve with a public key stored in
          an X.509v3 certificate and using the SHA2 family of
          hashing algorithms.";
       reference
         "RFC 6187: X.509v3 Certificates for Secure Shell
                    Authentication";
     }

     identity x509v3-ecdsa-sha2-nistp384 {
       base public-key-alg-base;
       if-feature "ssh-ecc and ssh-x509-certs and ssh-sha2";
       description
         "Elliptic Curve Digital Signature Algorithm (ECDSA)
          using the nistp384 curve with a public key stored in
          an X.509v3 certificate and using the SHA2 family of
          hashing algorithms.";
       reference
         "RFC 6187: X.509v3 Certificates for Secure Shell
                    Authentication";
     }

     identity x509v3-ecdsa-sha2-nistp521 {
       base public-key-alg-base;
       if-feature "ssh-ecc and ssh-x509-certs and ssh-sha2";
       description
```

```
         "Elliptic Curve Digital Signature Algorithm (ECDSA)
          using the nistp521 curve with a public key stored in
          an X.509v3 certificate and using the SHA2 family of
          hashing algorithms.";
       reference
         "RFC 6187: X.509v3 Certificates for Secure Shell
                    Authentication";
     }

     identity key-exchange-alg-base {
       description
         "Base identity used to identify key exchange algorithms.";
     }

     identity diffie-hellman-group14-sha1 {
       base key-exchange-alg-base;
       description
         "Diffie-Hellman key exchange with SHA-1 as HASH and
          Oakley Group 14 (2048-bit MODP Group).";
       reference
         "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
     }

     identity diffie-hellman-group-exchange-sha1 {
       base key-exchange-alg-base;
       if-feature "ssh-dh-group-exchange";
       description
         "Diffie-Hellman Group and Key Exchange with SHA-1 as HASH.";
       reference
         "RFC 4419: Diffie-Hellman Group Exchange for the
                    Secure Shell (SSH) Transport Layer Protocol";
     }

     identity diffie-hellman-group-exchange-sha256 {
       base key-exchange-alg-base;
       if-feature "ssh-dh-group-exchange and ssh-sha2";
       description
         "Diffie-Hellman Group and Key Exchange with SHA-256 as HASH.";
       reference
         "RFC 4419: Diffie-Hellman Group Exchange for the
                    Secure Shell (SSH) Transport Layer Protocol";
     }

     identity ecdh-sha2-nistp256 {
       base key-exchange-alg-base;
       if-feature "ssh-ecc and ssh-sha2";
       description
         "Elliptic Curve Diffie-Hellman (ECDH) key exchange using the
```

```
           nistp256 curve and the SHA2 family of hashing algorithms.";
        reference
          "RFC 5656: Elliptic Curve Algorithm Integration in the
                     Secure Shell Transport Layer";
      }

      identity ecdh-sha2-nistp384 {
        base key-exchange-alg-base;
        if-feature "ssh-ecc and ssh-sha2";
        description
          "Elliptic Curve Diffie-Hellman (ECDH) key exchange using the
           nistp384 curve and the SHA2 family of hashing algorithms.";
        reference
          "RFC 5656: Elliptic Curve Algorithm Integration in the
                     Secure Shell Transport Layer";
      }

      identity ecdh-sha2-nistp521 {
        base key-exchange-alg-base;
        if-feature "ssh-ecc and ssh-sha2";
        description
          "Elliptic Curve Diffie-Hellman (ECDH) key exchange using the
           nistp521 curve and the SHA2 family of hashing algorithms.";
        reference
          "RFC 5656: Elliptic Curve Algorithm Integration in the
                     Secure Shell Transport Layer";
      }

      identity encryption-alg-base {
        description
          "Base identity used to identify encryption algorithms.";
      }

      identity triple-des-cbc {
        base encryption-alg-base;
        description
          "Three-key 3DES in CBC mode.";
        reference
          "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
      }

      identity aes128-cbc {
        base encryption-alg-base;
        description
          "AES in CBC mode, with a 128-bit key.";
        reference
          "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
      }
```

```
     identity aes192-cbc {
       base encryption-alg-base;
       description
         "AES in CBC mode, with a 192-bit key.";
       reference
         "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
     }

     identity aes256-cbc {
       base encryption-alg-base;
       description
         "AES in CBC mode, with a 256-bit key.";
       reference
         "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
     }

     identity aes128-ctr {
       base encryption-alg-base;
       if-feature "ssh-ctr";
       description
         "AES in SDCTR mode, with 128-bit key.";
       reference
         "RFC 4344: The Secure Shell (SSH) Transport Layer Encryption
                    Modes";
     }

     identity aes192-ctr {
       base encryption-alg-base;
       if-feature "ssh-ctr";
       description
         "AES in SDCTR mode, with 192-bit key.";
       reference
         "RFC 4344: The Secure Shell (SSH) Transport Layer Encryption
                    Modes";
     }

     identity aes256-ctr {
       base encryption-alg-base;
       if-feature "ssh-ctr";
       description
         "AES in SDCTR mode, with 256-bit key.";
       reference
         "RFC 4344: The Secure Shell (SSH) Transport Layer Encryption
            Modes";
     }

     identity mac-alg-base {
       description
```

```
         "Base identity used to identify message authentication
          code (MAC) algorithms.";
     }

     identity hmac-sha1 {
       base mac-alg-base;
       description
         "HMAC-SHA1";
       reference
         "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
     }

     identity hmac-sha2-256 {
       base mac-alg-base;
       if-feature "ssh-sha2";
       description
         "HMAC-SHA2-256";
       reference
         "RFC 6668: SHA-2 Data Integrity Verification for the
                    Secure Shell (SSH) Transport Layer Protocol";
     }

     identity hmac-sha2-512 {
       base mac-alg-base;
       if-feature "ssh-sha2";
       description
         "HMAC-SHA2-512";
       reference
         "RFC 6668: SHA-2 Data Integrity Verification for the
                    Secure Shell (SSH) Transport Layer Protocol";
     }

     // Groupings

     grouping transport-params-grouping {
       description
         "A reusable grouping for SSH transport parameters.";
       reference
         "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
       container host-key {
         description
           "Parameters regarding host key.";
         leaf-list host-key-alg {
           type identityref {
             base public-key-alg-base;
           }
           ordered-by user;
           description
```

```
            "Acceptable host key algorithms in order of descending
             preference.  The configured host key algorithms should
             be compatible with the algorithm used by the configured
             private key.  Please see Section 5 of RFC XXXX for
             valid combinations.

             If this leaf-list is not configured (has zero elements)
             the acceptable host key algorithms are implementation-
             defined.";
          reference
            "RFC XXXX: YANG Groupings for SSH Clients and SSH Servers";
        }
      }
      container key-exchange {
        description
          "Parameters regarding key exchange.";
        leaf-list key-exchange-alg {
          type identityref {
            base key-exchange-alg-base;
          }
          ordered-by user;
          description
            "Acceptable key exchange algorithms in order of descending
             preference.

             If this leaf-list is not configured (has zero elements)
             the acceptable key exchange algorithms are implementation
             defined.";
        }
      }
      container encryption {
        description
          "Parameters regarding encryption.";
        leaf-list encryption-alg {
          type identityref {
            base encryption-alg-base;
          }
          ordered-by user;
          description
            "Acceptable encryption algorithms in order of descending
             preference.

             If this leaf-list is not configured (has zero elements)
             the acceptable encryption algorithms are implementation
             defined.";
        }
      }
      container mac {
```

```
         description
           "Parameters regarding message authentication code (MAC).";
         leaf-list mac-alg {
           type identityref {
             base mac-alg-base;
           }
           ordered-by user;
           description
             "Acceptable MAC algorithms in order of descending
              preference.

              If this leaf-list is not configured (has zero elements)
              the acceptable MAC algorithms are implementation-
              defined.";
         }
       }
     }
   }
   <CODE ENDS>
```

6. Security Considerations

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

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

Since the modules in this document only define groupings, these considerations are primarily for the designers of other modules that use these groupings.

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

*: All of the nodes defined by the grouping statement in both the "ietf-ssh-client" and "ietf-ssh-server" modules are sensitive to write operations. For instance, the addition or removal of references to keys, certificates, trusted anchors, etc., or

even the modification of transport or keepalive parameters can dramatically alter the implemented security policy. For this reason, all the nodes are protected the NACM extension "default-deny-write".

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

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

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

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

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

7. IANA Considerations

7.1. The IETF XML Registry

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

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

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

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

7.2. The YANG Module Names Registry

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

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

name:         ietf-ssh-server
namespace:    urn:ietf:params:xml:ns:yang:ietf-ssh-server
prefix:       sshs
reference:    RFC XXXX

name:         ietf-ssh-common
namespace:    urn:ietf:params:xml:ns:yang:ietf-ssh-common
prefix:       sshcmn
reference:    RFC XXXX
```

8. References

8.1. Normative References

[I-D.ietf-netconf-crypto-types]
Watsen, K. and H. Wang, "Common YANG Data Types for Cryptography", draft-ietf-netconf-crypto-types-06 (work in progress), April 2019.

[I-D.ietf-netconf-keystore]
Watsen, K., "YANG Data Model for a Centralized Keystore Mechanism", draft-ietf-netconf-keystore-09 (work in progress), April 2019.

[I-D.ietf-netconf-trust-anchors]
Watsen, K., "YANG Data Model for Global Trust Anchors", draft-ietf-netconf-trust-anchors-04 (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>.

[RFC4344] Bellare, M., Kohno, T., and C. Namprempre, "The Secure Shell (SSH) Transport Layer Encryption Modes", RFC 4344, DOI 10.17487/RFC4344, January 2006, <https://www.rfc-editor.org/info/rfc4344>.

[RFC4419] Friedl, M., Provos, N., and W. Simpson, "Diffie-Hellman Group Exchange for the Secure Shell (SSH) Transport Layer Protocol", RFC 4419, DOI 10.17487/RFC4419, March 2006, <https://www.rfc-editor.org/info/rfc4419>.

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

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

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

[RFC6668] Bider, D. and M. Baushke, "SHA-2 Data Integrity Verification for the Secure Shell (SSH) Transport Layer Protocol", RFC 6668, DOI 10.17487/RFC6668, July 2012, <https://www.rfc-editor.org/info/rfc6668>.

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

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

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

8.2. Informative References

[OPENSSH] Project, T. O., "OpenSSH", 2016, <http://www.openssh.com>.

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

[RFC4252] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Authentication Protocol", RFC 4252, DOI 10.17487/RFC4252, January 2006, <https://www.rfc-editor.org/info/rfc4252>.

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

[RFC4254] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Connection Protocol", RFC 4254, DOI 10.17487/RFC4254, January 2006, <https://www.rfc-editor.org/info/rfc4254>.

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

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

[RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, DOI 10.17487/RFC7317, August 2014, <https://www.rfc-editor.org/info/rfc7317>.

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

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

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

Appendix A. Change Log

A.1. 00 to 01

- o Noted that '0.0.0.0' and '::' might have special meanings.

- o Renamed "keychain" to "keystore".

A.2. 01 to 02

- o Removed the groupings 'listening-ssh-client-grouping' and 'listening-ssh-server-grouping'. Now modules only contain the transport-independent groupings.

- o Simplified the "client-auth" part in the ietf-ssh-client module. It now inlines what it used to point to keystore for.

- o Added cipher suites for various algorithms into new 'ietf-ssh-common' module.

A.3. 02 to 03

- o Removed 'RESTRICTED' enum from 'password' leaf type.

- o Added a 'must' statement to container 'server-auth' asserting that at least one of the various auth mechanisms must be specified.

- o Fixed description statement for leaf 'trusted-ca-certs'.

A.4. 03 to 04

- o Change title to "YANG Groupings for SSH Clients and SSH Servers"

- o Added reference to RFC 6668

- o Added RFC 8174 to Requirements Language Section.

- o Enhanced description statement for ietf-ssh-server's "trusted-ca-certs" leaf.

- o Added mandatory true to ietf-ssh-client's "client-auth" 'choice' statement.

- o Changed the YANG prefix for module ietf-ssh-common from 'sshcom' to 'sshcmn'.

- o Removed the compression algorithms as they are not commonly configurable in vendors' implementations.

- Updating descriptions in transport-params-grouping and the servers's usage of it.

- Now tree diagrams reference ietf-netmod-yang-tree-diagrams

- Updated YANG to use typedefs around leafrefs to common keystore paths

- Now inlines key and certificates (no longer a leafref to keystore)

A.5. 04 to 05

- Merged changes from co-author.

A.6. 05 to 06

- Updated to use trust anchors from trust-anchors draft (was keystore draft)

- Now uses new keystore grouping enabling asymmetric key to be either locally defined or a reference to the keystore.

A.7. 06 to 07

- factored the ssh-[client|server]-groupings into more reusable groupings.

- added if-feature statements for the new "ssh-host-keys" and "x509-certificates" features defined in draft-ietf-netconf-trust-anchors.

A.8. 07 to 08

- Added a number of compatibility matrices to Section 5 (thanks Frank!)

- Clarified that any configured "host-key-alg" values need to be compatible with the configured private key.

A.9. 08 to 09

- Updated examples to reflect update to groupings defined in the keystore -09 draft.

- Add SSH keepalives features and groupings.

- Prefixed top-level SSH grouping nodes with 'ssh-' and support mashups.

- o Updated copyright date, boilerplate template, affiliation, and folding algorithm.

A.10. 09 to 10

- o Reformatted the YANG modules.

A.11. 10 to 11

- o Reformatted lines causing folding to occur.

A.12. 11 to 12

- o Collapsed all the inner groupings into the top-level grouping.

- o Added a top-level "demux container" inside the top-level grouping.

- o Added NACM statements and updated the Security Considerations section.

- o 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.

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

A.13. 12 to 13

- o Removed the "demux containers", floating the nacm:default-deny-write to each descendent node, and adding a note to model designers regarding the potential need to add their own demux containers.

- o Fixed a couple references (section 2 --> section 3)

- o In the server model, replaced <client-cert-auth> with <client-authentication> and introduced 'local-or-external' choice.

A.14. 13 to 14

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

Acknowledgements

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

Authors' Addresses

Kent Watsen
Watsen Networks

EMail: kent+ietf@watsen.net

Gary Wu
Cisco Systems

EMail: garywu@cisco.com

Liang Xia
Huawei

EMail: frank.xialiang@huawei.com

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

Copyright Notice

Table of Contents

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 extensibly so that they may be augmented for subscription targets beyond event streams. For examples of such augmentations, see the RPC augmentations within [I-D.ietf-netconf-yang-push]'s YANG model.

2.4.1. Dynamic Subscription State Model

Below is the publisher's state machine for a dynamic subscription. Each state is shown in its own box. It is important to note that such a subscription doesn't exist at the publisher until an "establish-subscription" RPC is accepted. The mere request by a subscriber to establish a subscription is insufficient for that subscription to be externally visible. Start and end states are depicted to reflect subscription creation and deletion events.

```
                     .........
                     : start :
                     :.......:
                         |
                establish-subscription
                         |
                         |   .-------modify-subscription--------.
                         v   v                                  |
                   .-----------.                          .-----------.
        .--------. | receiver  |--insufficient CPU, b/w-->| receiver  |
    modify-       '|  active   |                          | suspended |
    subscription   |           |<----CPU, b/w sufficient--|           |
        ---------->'-----------'                          '-----------'
                         |                                      |
              delete/kill-subscription                     delete/kill-
                         |                                 subscription
                         v                                      |
                     .........                                  |
                     :  end  :<---------------------------------'
                     :.......:
```

Figure 1: Publisher's state for a dynamic subscription

Of interest in this state machine are the following:

- o Successful "establish-subscription" or "modify-subscription" RPCs put the subscription into the active state.
- o Failed "modify-subscription" RPCs will leave the subscription in its previous state, with no visible change to any streaming updates.
- o A "delete-subscription" or "kill-subscription" RPC will end the subscription, as will the reaching of a "stop-time".
- o A publisher may choose to suspend a subscription when there is insufficient CPU or bandwidth available to service the

subscription. This is notified to a subscriber with a "subscription-suspended" subscription state change notification.

o A suspended subscription may be modified by the subscriber (for example in an attempt to use fewer resources). Successful modification returns the subscription to the active state.

o Even without a "modify-subscription" request, a publisher may return a subscription to the active state should the resource constraints become sufficient again. This is announced to the subscriber via the "subscription-resumed" subscription state change notification.

2.4.2. Establishing a Dynamic Subscription

The "establish-subscription" RPC allows a subscriber to request the creation of a subscription.

The input parameters of the operation are:

o A "stream" name which identifies the targeted event stream against which the subscription is applied.

o An event stream filter which may reduce the set of event records pushed.

o Where the transport used by the RPC supports multiple encodings, an optional "encoding" for the event records pushed. If no "encoding" is included, the encoding of the RPC MUST be used.

o An optional "stop-time" for the subscription. If no "stop-time" is present, notification messages will continue to be sent until the subscription is terminated.

o An optional "replay-start-time" for the subscription. The "replay-start-time" MUST be in the past and indicates that the subscription is requesting a replay of previously generated information from the event stream. For more on replay, see Section 2.4.2.1. Where there is no "replay-start-time", the subscription starts immediately.

If the publisher can satisfy the "establish-subscription" request, it replies with an identifier for the subscription, and then immediately starts streaming notification messages.

Below is a tree diagram for "establish-subscription". All objects contained in this tree are described within the included YANG model within Section 4.

```
      +---x establish-subscription
         +---w input
         |  +---w (target)
         |  |  +--:(stream)
         |  |     +---w (stream-filter)?
         |  |     |  +--:(by-reference)
         |  |     |  |  +---w stream-filter-name
         |  |     |  |          stream-filter-ref
         |  |     |  +--:(within-subscription)
         |  |     |     +---w (filter-spec)?
         |  |     |        +--:(stream-subtree-filter)
         |  |     |        |  +---w stream-subtree-filter?   <anydata>
         |  |     |        |          {subtree}?
         |  |     |        +--:(stream-xpath-filter)
         |  |     |           +---w stream-xpath-filter?
         |  |     |                   yang:xpath1.0 {xpath}?
         |  |     +---w stream                               stream-ref
         |  |     +---w replay-start-time?
         |  |             yang:date-and-time {replay}?
         |  +---w stop-time?
         |  |       yang:date-and-time
         |  +---w dscp?                                      inet:dscp
         |  |       {dscp}?
         |  +---w weighting?                                 uint8
         |  |       {qos}?
         |  +---w dependency?
         |  |       subscription-id {qos}?
         |  +---w encoding?                                  encoding
         +--ro output
            +--ro id                             subscription-id
            +--ro replay-start-time-revision?   yang:date-and-time
                    {replay}?
```

Figure 2: establish-subscription RPC tree diagram

A publisher MAY reject the "establish-subscription" RPC for many reasons as described in Section 2.4.6. The contents of the resulting RPC error response MAY include details on input parameters which if considered in a subsequent "establish-subscription" RPC, may result in a successful subscription establishment. Any such hints MUST be transported within a yang-data "establish-subscription-stream-error-info" container included within the RPC error response.

```
yang-data establish-subscription-stream-error-info
   +--ro establish-subscription-stream-error-info
      +--ro reason?                 identityref
      +--ro filter-failure-hint?    string
```

Figure 3: establish-subscription RPC yang-data tree diagram

2.4.2.1. Requesting a replay of event records

Replay provides the ability to establish a subscription which is also capable of passing 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.

```
 .........
 : start :-.
 :.......: |
      create  .---modify-----.----------------------------------.
           |  |              |                                  |
           V  V          .-------.         .......         .---------.
  .----[evaluate]--no--->|invalid|-delete->: end :<-delete-|concluded|
  |                      '-------'         :.....:         '---------'
  |-[evaluate]--no-(2).      ^                ^                 ^
  |        ^          |      |                |                 |
 yes       |          '->unsupportable      delete           stop-time
  |      modify         (subscription-   (subscription-   (subscription-
  |        |             terminated*)     terminated*)      concluded*)
  |        |                 |                |                 |
 (1)       |                (3)              (4)               (5)
  |   .---------------------------------------------------------------.
  '-->|                            valid                              |
      '---------------------------------------------------------------'

 Legend:
 dotted boxes: subscription added or removed via configuration
 dashed boxes: states for a subscription
 [evaluate]: decision point on whether the subscription is supportable
 (*): resulting subscription state change notification
```

Figure 8: Publisher state model for a configured subscription

A subscription in the valid state may move to the invalid state in one of two ways. First, it may be modified in a way which fails a re-evaluation. See (2) in the diagram. Second, the publisher might determine that the subscription is no longer supportable. This could be for reasons of an unexpected but sustained increase in an event stream's event records, degraded CPU capacity, a more complex referenced filter, or other subscriptions which have usurped resources. See (3) in the diagram. No matter the case, a "subscription-terminated" notification is sent to any receivers in an active or suspended state. A subscription in the valid state may also transition to the concluded state via (5) if a configured stop time has been reached. In this case, a "subscription-concluded" notification is sent to any receivers in active or suspended states. Finally, a subscription may be deleted by configuration (4).

When a subscription is in the valid state, a publisher will attempt to connect with all receivers of a configured subscription and deliver notification messages. Below is the state machine for each receiver of a configured subscription. This receiver state machine is fully contained within the state machine of the configured subscription, and is only relevant when the configured subscription is in the valid state.

```
  .-----------------------------------------------------------------.
  |                         valid                                   |
  |   .----------.                           .------------.         |
  |   | receiver |---timeout---------------->|  receiver  |         |
  |   |connecting|<----------------reset--(c)|disconnected|         |
  |   |          |<-transport                '------------'         |
  |   '----------'  loss,reset------------------------------.       |
  |      (a)          |                                     |       |
  |  subscription-   (b)                                   (b)      |
  |  started*    .--------.                             .---------. |
  |       '----->|        |(d)-insufficient CPU,------->|         | |
  |              |receiver|    buffer overflow          |receiver | |
  | subscription-| active |                             |suspended| |
  |   modified*  |        |<----CPU, b/w sufficient,-(e)|         | |
  |        '---->'--------'     subscription-modified*  '---------' |
  '-----------------------------------------------------------------'
```

Legend:
dashed boxes which include the word 'receiver' show the possible states for an individual receiver of a valid configured subscription.
* indicates a subscription state change notification

Figure 9: Receiver state for a configured subscription on a Publisher

When a configured subscription first moves to the valid state, the "state" leaf of each receiver is initialized to the connecting state. If transport connectivity is not available to any receiver and there are any notification messages to deliver, a transport session is established (e.g., through [RFC8071]). Individual receivers are moved to the active state when a "subscription-started" subscription state change notification is successfully passed to that receiver (a). Event records are only sent to active receivers. Receivers of a configured subscription remain active if both transport connectivity can be verified to the receiver, and event records are not being dropped due to a publisher buffer 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.

```
+---n subscription-modified
   +--ro id
   |       subscription-id
   +--ro (target)
   |  +--:(stream)
   |     +--ro (stream-filter)?
   |     |  +--:(by-reference)
   |     |  |  +--ro stream-filter-name
   |     |  |          stream-filter-ref
   |     |  +--:(within-subscription)
   |     |     +--ro (filter-spec)?
   |     |        +--:(stream-subtree-filter)
   |     |        |  +--ro stream-subtree-filter?   <anydata>
   |     |        |          {subtree}?
   |     |        +--:(stream-xpath-filter)
   |     |           +--ro stream-xpath-filter?     yang:xpath1.0
   |     |                   {xpath}?
   |     +--ro stream                               stream-ref
   |     +--ro replay-start-time?
   |             yang:date-and-time {replay}?
   +--ro stop-time?
   |       yang:date-and-time
   +--ro dscp?                                      inet:dscp
   |       {dscp}?
   +--ro weighting?                                 uint8 {qos}?
   +--ro dependency?
   |       subscription-id {qos}?
   +--ro transport?                                 transport
   |       {configured}?
   +--ro encoding?                                  encoding
   +--ro purpose?                                   string
           {configured}?
```

Figure 12: subscription-modified notification tree diagram

A publisher most often sends this notification directly after the modification of any configuration parameters impacting a configured subscription. But it may also be sent at two other times:

1. Where a configured subscription has been modified during the suspension of a receiver, the notification will be delayed until the receiver's suspension is lifted. In this situation, the notification indicates that the subscription has been both modified and resumed.

2. A "subscription-modified" subscription state change notification MUST be sent if the contents of the filter identified by the subscription's "stream-filter-ref" leaf has changed. This state change notification is to be sent for a filter change impacting any active receiver of a configured or dynamic subscription.

2.7.3. subscription-terminated

This notification indicates that no further event records for this subscription should be expected from the publisher. A publisher may terminate the sending event records to a receiver for the following reasons:

1. Configuration which removes a configured subscription, or a "kill-subscription" RPC which ends a dynamic subscription. These are identified via the reason "no-such-subscription".

2. A referenced filter is no longer accessible. This is identified by "filter-unavailable".

3. The event stream referenced by a subscription is no longer accessible by the receiver. This is identified by "stream-unavailable".

4. A suspended subscription has exceeded some timeout. This is identified by "suspension-timeout".

Each of the reasons above correspond one-to-one with a "reason" identityref specified within the YANG model.

Below is a tree diagram for "subscription-terminated". All objects contained in this tree are described within the included YANG model within Section 4.

```
+---n subscription-terminated
   +--ro id        subscription-id
   +--ro reason    identityref
```

Figure 13: subscription-terminated notification tree diagram

Note: this subscription state change notification MUST be sent to a dynamic subscription's receiver when the subscription ends unexpectedly. The cases when this might happen are when a "kill-subscription" RPC is successful, or when some other event not including the reaching the subscription's "stop-time" results in a publisher choosing to end the subscription.

2.7.4. subscription-suspended

This notification indicates that a publisher has suspended the sending of event records to a receiver, and also indicates the possible loss of events. Suspension happens when capacity constraints stop a publisher from serving a valid subscription. The two conditions where is this possible are:

1. "insufficient-resources" when a publisher is unable to produce the requested event stream of notification messages, and

2. "unsupportable-volume" when the bandwidth needed to get generated notification messages to a receiver exceeds a threshold.

These conditions are encoded within the "reason" object. No further notification will be sent until the subscription resumes or is terminated.

Below is a tree diagram for "subscription-suspended". All objects contained in this tree are described within the included YANG model within Section 4.

```
+---n subscription-suspended
   +--ro id        subscription-id
   +--ro reason    identityref
```

Figure 14: subscription-suspended notification tree diagram

2.7.5. subscription-resumed

This notification indicates that a previously suspended subscription has been resumed under the unmodified terms previously in place. Subscribed event records generated after the issuance of this subscription state change notification may now be sent.

Below is the tree diagram for "subscription-resumed". All objects contained in this tree are described within the included YANG model within Section 4.

```
+---n subscription-resumed
   +--ro id    subscription-id
```

Figure 15: subscription-resumed notification tree diagram

2.7.6. subscription-completed

This notification indicates that a subscription that includes a "stop-time" has successfully finished passing event records upon the reaching of that time.

Below is a tree diagram for "subscription-completed". All objects contained in this tree are described within the included YANG model within Section 4.

```
    +---n subscription-completed {configured}?
       +--ro id    subscription-id
```

Figure 16: subscription-completed notification tree diagram

2.7.7. replay-completed

This notification indicates that all of the event records prior to the current time have been passed to a receiver. It is sent before any notification message containing an event record with a timestamp later than (1) the "stop-time" or (2) the subscription's start time.

If a subscription contains no "stop-time", or has a "stop-time" that has not been reached, then after the "replay-completed" notification has been sent, additional event records will be sent in sequence as they arise naturally on the publisher.

Below is a tree diagram for "replay-completed". All objects contained in this tree are described within the included YANG model within Section 4.

```
    +---n replay-completed {replay}?
       +--ro id    subscription-id
```

Figure 17: replay-completed notification tree diagram

2.8. Subscription Monitoring

In the operational state datastore, the container "subscriptions" maintains the state of all dynamic subscriptions, as well as all configured subscriptions. Using datastore retrieval operations, or subscribing to the "subscriptions" container [I-D.ietf-netconf-yang-push] allows the state of subscriptions and their connectivity to receivers to be monitored.

Each subscription in the operational state datastore is represented as a list element. Included in this list are event counters for each receiver, the state of each receiver, as well as the subscription parameters currently in effect. The appearance of the leaf "configured-subscription-state" indicates that a particular subscription came into being via configuration. This leaf also indicates if the current state of that subscription is valid, invalid, and concluded.

To understand the flow of event records within a subscription, there are two counters available for each receiver. The first counter is "sent-event-records" which shows the quantity of events actually identified for sending to a receiver. The second counter is "excluded-event-records" which shows event records not sent to receiver. "excluded-event-records" shows the combined results of both access control and per-subscription filtering. For configured subscriptions, counters are reset whenever the subscription is evaluated to valid (see (1) in Figure 8).

Dynamic subscriptions are removed from the operational state datastore once they expire (reaching stop-time) or when they are terminated. While many subscription objects are shown as configurable, dynamic subscriptions are only included within the operational state datastore and as a result are not configurable.

2.9. Advertisement

Publishers supporting this document MUST indicate support of the YANG model "ietf-subscribed-notifications" within the YANG library of the publisher. In addition if supported, the optional features "encode-xml", "encode-json", "configured" "supports-vrf", "qos", "xpath", "subtree", "interface-designation", "dscp", and "replay" MUST be indicated.

3. YANG Data Model Trees

This section contains tree diagrams for nodes defined in Section 4. For tree diagrams of subscription state change notifications, see Section 2.7. For the tree diagrams for the RPCs, see Section 2.4.

3.1. Event Streams Container

A publisher maintains a list of available event streams as operational data. This list contains both standardized and vendor-specific event streams. This enables subscribers to discover what streams a publisher supports.

```
  +--ro streams
     +--ro stream* [name]
        +--ro name                          string
        +--ro description                   string
        +--ro replay-support?               empty {replay}?
        +--ro replay-log-creation-time      yang:date-and-time
        |       {replay}?
        +--ro replay-log-aged-time?         yang:date-and-time
                {replay}?
```

Figure 18: Stream Container tree diagram

Above is a tree diagram for the "streams" container. All objects contained in this tree are described within the included YANG model within Section 4.

3.2. Filters Container

The "filters" container maintains a list of all subscription filters that persist outside the life-cycle of a single subscription. This enables pre-defined filters which may be referenced by more than one subscription.

```
  +--rw filters
     +--rw stream-filter* [name]
        +--rw name                             string
        +--rw (filter-spec)?
           +--:(stream-subtree-filter)
           |  +--rw stream-subtree-filter?   <anydata> {subtree}?
           +--:(stream-xpath-filter)
              +--rw stream-xpath-filter?     yang:xpath1.0 {xpath}?
```

Figure 19: Filter Container tree diagram

Above is a tree diagram for the filters container. All objects contained in this tree are described within the included YANG model within Section 4.

3.3. Subscriptions Container

The "subscriptions" container maintains a list of all subscriptions on a publisher, both configured and dynamic. It can be used to retrieve information about the subscriptions which a publisher is serving.

```
  +--rw subscriptions
```

```
        +--rw subscription* [id]
           +--rw id
           |       subscription-id
           +--rw (target)
           |  +--:(stream)
           |     +--rw (stream-filter)?
           |     |  +--:(by-reference)
           |     |  |  +--rw stream-filter-name
           |     |  |          stream-filter-ref
           |     |  +--:(within-subscription)
           |     |     +--rw (filter-spec)?
           |     |        +--:(stream-subtree-filter)
           |     |        |  +--rw stream-subtree-filter?   <anydata>
           |     |        |          {subtree}?
           |     |        +--:(stream-xpath-filter)
           |     |           +--rw stream-xpath-filter?
           |     |                   yang:xpath1.0 {xpath}?
           |     +--rw stream                               stream-ref
           |     +--ro replay-start-time?
           |     |       yang:date-and-time {replay}?
           |     +--rw configured-replay?                   empty
           |             {configured,replay}?
           +--rw stop-time?
           |       yang:date-and-time
           +--rw dscp?                                      inet:dscp
           |       {dscp}?
           +--rw weighting?                                 uint8 {qos}?
           +--rw dependency?
           |       subscription-id {qos}?
           +--rw transport?                                 transport
           |       {configured}?
           +--rw encoding?                                  encoding
           +--rw purpose?                                   string
           |       {configured}?
           +--rw (notification-message-origin)? {configured}?
           |  +--:(interface-originated)
           |  |  +--rw source-interface?
           |  |          if:interface-ref {interface-designation}?
           |  +--:(address-originated)
           |     +--rw source-vrf?
           |     |       -> /ni:network-instances/network-instance/name
           |     |       {supports-vrf}?
           |     +--rw source-address?
           |             inet:ip-address-no-zone
           +--ro configured-subscription-state?           enumeration
           |       {configured}?
           +--rw receivers
              +--rw receiver* [name]
```

```
                 +--rw name                      string
                 +--ro sent-event-records?
                 |       yang:zero-based-counter64
                 +--ro excluded-event-records?
                 |       yang:zero-based-counter64
                 +--ro state                     enumeration
                 +---x reset {configured}?
                    +--ro output
                       +--ro time    yang:date-and-time
```

Figure 20: Subscriptions tree diagram

Above is a tree diagram for the subscriptions container. All objects contained in this tree are described within the included YANG model within Section 4.

4. Data Model

This module imports typedefs from [RFC6991], [RFC8343], and [RFC8040], and it references [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 augmentated to a specific receiver
            within this list.";
          leaf name {
            type string;
            description
              "Identifies a unique receiver for a subscription.";
          }
          leaf sent-event-records {
            type yang:zero-based-counter64;
            config false;
            description
              "The number of event records sent to the receiver.  The
```

```
               count is initialized when a dynamic subscription is
               established, or when a configured receiver
               transitions to the valid state.";
           }
           leaf excluded-event-records {
             type yang:zero-based-counter64;
             config false;
             description
               "The number of event records explicitly removed either
               via an event stream filter or an access control filter so
               that they are not passed to a receiver.  This count is
               set to zero each time 'sent-event-records' is
               initialized.";
           }
           leaf state {
             type enumeration {
               enum active {
                 value 1;
                 description
                   "Receiver is currently being sent any applicable
                   notification messages for the subscription.";
               }
               enum suspended {
                 value 2;
                 description
                   "Receiver state is 'suspended', so the publisher
                   is currently unable to provide notification messages
                   for the subscription.";
               }
               enum connecting {
                 value 3;
                 if-feature "configured";
                 description
                   "A subscription has been configured, but a
                   'subscription-started' subscription state change
                   notification needs to be successfully received before
                   notification messages are sent.

                   If the 'reset' action is invoked for a receiver of an
                   active configured subscription, the state must be
                   moved to 'connecting'.";
               }
               enum disconnected {
                 value 4;
                 if-feature "configured";
                 description
                   "A subscription has failed in sending a subscription
                   started state change to the receiver.
```

```
                   Additional attempts at connection attempts are not
                   currently being made.";
               }
             }
             config false;
             mandatory true;
             description
               "Specifies the state of a subscription from the
               perspective of a particular receiver.  With this info it
               is possible to determine whether a 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 identity 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.

- "source-address": the configured address might not be able to reach a desired receiver.

- "source-interface": the configured interface might not be able to reach a desired receiver.

- "source-vrf": can place a subscription into a virtual network where receivers are not entitled to view the subscribed content.

- "stop-time": could be used to terminate content at an inopportune time.

- "stream": could set a subscription to an event stream containing no content permitted for the targeted receivers.

- "stream-filter-name": could be set to a filter which is irrelevant to the event stream.

- "stream-subtree-filter": a complex filter can increase the computational resources for this subscription.

- "stream-xpath-filter": a complex filter can increase the computational resources for this subscription.

- "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"

- "name": if access control is not properly configured, can expose system internals to those who should have no access to this information.

- "replay-support": if access control is not properly configured, can expose logs to those who should have no access.

Container: "/subscriptions"

- "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.

- "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

- 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"

- No special considerations.

RPC: "establish-subscription"

- 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"

- The "kill-subscription" RPC MUST be secured so that only connections with administrative rights are able to invoke this RPC.

RPC: "modify-subscription"

- 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 claifications 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 were deltas existed

- o new feature for interface originated.

- o dscp removed from the qos feature

- o YANG model updated in a way which collapses groups only used once so that they are part of the 'subscriptions' container.

- o alternative encodings only allowed for transports which support them.

v09 - v10

- o Typos and tweaks

v08 - v09

- o NMDA model supported. Non NMDA version at https://github.com/netconf-wg/rfc5277bis/

- o Error mechanism revamped to match to embedded implementations.

- o Explicitly identified error codes relevant to each RPC/Notification

v07 - v08

- o Split YANG trees to separate document subsections.

- o Clarified configured state machine based on Balazs comments, and moved it into the configured subscription subsections.

- o Normative reference to Network Instance model for VRF

- o One transport for all receivers of configured subscriptions.

- o QoS section moved in from yang-push

v06 - v07

o Clarification on state machine for configured subscriptions.

v05 - v06

o Made changes proposed by Martin, Kent, and others on the list. Most significant of these are stream returned to string (with the SYSLOG identity removed), intro section on 5277 relationship, an identity set moved to an enumeration, clean up of definitions/ terminology, state machine proposed for configured subscriptions with a clean-up of subscription state options.

o JSON and XML become features. Also Xpath and subtree filtering become features

o Terminology updates with event records, and refinement of filters to just event stream filters.

o Encoding refined in establish-subscription so it takes the RPC's encoding as the default.

o Namespaces in examples fixed.

v04 - v05

o Returned to the explicit filter subtyping of v00

o stream object changed to 'name' from 'stream'

o Cleaned up examples

o Clarified that JSON support needs notification-messages draft.

v03 - v04

o Moved back to the use of RFC5277 one-way notifications and encodings.

v03 - v04

o Replay updated

v02 - v03

o RPCs and Notification support is identified by the Notification 2.0 capability.

- o Updates to filtering identities and text
- o New error type for unsupportable volume of updates
- o Text tweaks.

v01 - v02

- o Subscription status moved under receiver.

v00 - v01

- o Security considerations updated
- o Intro rewrite, as well as scattered text changes
- o Added Appendix A, to help match this to related drafts in progress
- o Updated filtering definitions, and filter types in yang file, and moved to identities for filter types
- o Added Syslog as an event stream
- o HTTP2 moved in from YANG-Push as a transport option
- o Replay made an optional feature for events. Won't apply to datastores
- o Enabled notification timestamp to have different formats.
- o Two error codes added.

v01 5277bis - v00 subscribed notifications

- o Kill subscription RPC added.
- o Renamed from 5277bis to Subscribed Notifications.
- o Changed the notification capabilities version from 1.1 to 2.0.
- o Extracted create-subscription and other elements of RFC5277.
- o Error conditions added, and made specific in return codes.
- o Simplified yang model structure for removal of 'basic' grouping.
- o Added a grouping for items which cannot be statically configured.

- Operational counters per receiver.
- Subscription-id and filter-id renamed to identifier
- Section for replay added. Replay now cannot be configured.
- Control plane notification renamed to subscription state change notification
- Source address: Source-vrf changed to string, default address option added
- In yang model: 'info' changed to 'policy'
- Scattered text clarifications

v00 - v01 of 5277bis

- 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.
- Expanded/renamed definitions from event server to publisher, and client to subscriber as applicable. Updated the definitions to include and expand on RFC 5277.
- Removal of redundancy with other drafts
- Many other clean-ups of wording and terminology

Authors' Addresses

Eric Voit
Cisco Systems

Email: evoit@cisco.com

Alexander Clemm
Huawei

Email: ludwig@clemm.org

Alberto Gonzalez Prieto
Microsoft

Email: alberto.gonzalez@microsoft.com

Einar Nilsen-Nygaard
Cisco Systems

Email: einarnn@cisco.com

Ambika Prasad Tripathy
Cisco Systems

Email: ambtripa@cisco.com

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2020

K. Watsen
Watsen Networks
G. Wu
Cisco Systems
L. Xia
Huawei

July 2, 2019

YANG Groupings for TLS Clients and TLS Servers
draft-ietf-netconf-tls-client-server-14

Abstract

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

Editorial Note (To be removed by RFC Editor)

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

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

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

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

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

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

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

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

o Appendix A. Change Log

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document defines three YANG 1.1 [RFC7950] modules: the first defines a grouping for a generic TLS client, the second defines a grouping for a generic TLS server, and the third defines identities and groupings common to both the client and the server (TLS is defined in [RFC5246]). It is intended that these groupings will be used by applications using the TLS protocol. For instance, these groupings could be used to help define the data model for an HTTPS [RFC2818] server or a NETCONF over TLS [RFC7589] based server.

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

2. Terminology

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

3. The TLS Client Model

3.1. Tree Diagram

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

```
module: ietf-tls-client

  grouping tls-client-grouping
    +-- client-identity
    |  +---u ks:local-or-keystore-end-entity-cert-with-key-grouping
    +-- server-authentication
    |  +-- ca-certs?       ts:certificates-ref
    |  |       {ts:x509-certificates}?
    |  +-- server-certs?   ts:certificates-ref
    |          {ts:x509-certificates}?
    +-- hello-params {tls-client-hello-params-config}?
    |  +---u tlscmn:hello-params-grouping
    +-- keepalives! {tls-client-keepalives}?
       +-- max-wait?       uint16
       +-- max-attempts?   uint8
```

3.2. Example Usage

This section presents two examples showing the tls-client-grouping populated with some data. These examples are effectively the same except the first configures the client identity using a local key while the second uses a key configured in a keystore. Both examples are consistent with the examples presented in Section 2 of [I-D.ietf-netconf-trust-anchors] and Section 3.2 of [I-D.ietf-netconf-keystore].

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

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

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

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

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

</tls-client>
```


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

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

  <!-- how this client will authenticate itself to the server -->
  <client-identity>
    <keystore-reference>
      <asymmetric-key>ex-rsa-key</asymmetric-key>
      <certificate>ex-rsa-cert</certificate>
    </keystore-reference>
  </client-identity>

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

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

</tls-client>
```

3.3. YANG Module

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

```
<CODE BEGINS> file "ietf-tls-client@2019-07-02.yang"
module ietf-tls-client {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-tls-client";
  prefix tlsc;

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

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

  import ietf-keystore {
```

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

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

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

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

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

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

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

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

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

  revision 2019-07-02 {
    description
      "Initial version";
```

```
       reference
         "RFC XXXX: YANG Groupings for TLS Clients and TLS Servers";
     }

     // Features

     feature tls-client-hello-params-config {
       description
         "TLS hello message parameters are configurable on a TLS
          client.";
     }

     feature tls-client-keepalives {
       description
         "Per socket TLS keepalive parameters are configurable for
          TLS clients on the server implementing this feature.";
     }

     // Groupings

     grouping tls-client-grouping {
       description
         "A reusable grouping for configuring a TLS client without
          any consideration for how an underlying TCP session is
          established.

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

       container client-identity { // FIXME: what about PSKs?
         nacm:default-deny-write;
         description
           "A locally-defined or referenced end-entity certificate,
            including any configured intermediate certificates, the
            TLS client will present when establishing a TLS connection
            in its Certificate message, as defined in Section 7.4.2
            in RFC 5246.";
         reference
           "RFC 5246:
              The Transport Layer Security (TLS) Protocol Version 1.2
            RFC ZZZZ:
              YANG Data Model for a 'Keystore' Mechanism";
```

```
          uses ks:local-or-keystore-end-entity-cert-with-key-grouping;
        } // container client-identity

        container server-authentication {  // FIXME: what about PSKs?
          nacm:default-deny-write;
          must 'ca-certs or server-certs';
          description
            "Trusted server identities.";
          leaf ca-certs {
            if-feature "ts:x509-certificates";
            type ts:certificates-ref;
            description
              "A reference to a list of certificate authority (CA)
               certificates used by the TLS client to authenticate
               TLS server certificates.  A server certificate is
               authenticated if it has a valid chain of trust to
               a configured CA certificate.";
          }
          leaf server-certs {
            if-feature "ts:x509-certificates";
            type ts:certificates-ref;
            description
              "A reference to a list of server certificates used by
               the TLS client to authenticate TLS server certificates.
               A server certificate is authenticated if it is an
               exact match to a configured server certificate.";
          }
        } // container server-authentication

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

        container keepalives {
          nacm:default-deny-write;
          if-feature "tls-client-keepalives";
          presence "Indicates that keepalives are enabled.";
          description
            "Configures the keep-alive policy, to proactively test
             the aliveness of the TLS server.  An unresponsive
             TLS server is dropped after approximately max-wait
             * max-attempts seconds.";
          leaf max-wait {
            type uint16 {
```

```
             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.";
         }
       } // container keepalives
     } // grouping tls-client-grouping
   }
   <CODE ENDS>
```

4. The TLS Server Model

4.1. Tree Diagram

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

```
   module: ietf-tls-server

     grouping tls-server-grouping
       +-- server-identity
       |  +---u ks:local-or-keystore-end-entity-cert-with-key-grouping
       +-- client-authentication!
       |  +-- (required-or-optional)
       |  |  +--:(required)
       |  |  |  +-- required?                        empty
       |  |  +--:(optional)
       |  |     +-- optional?                        empty
       |  +-- (local-or-external)
       |     +--:(local) {local-client-auth-supported}?
       |     |  +-- ca-certs?                        ts:certificates-ref
       |     |  |       {ts:x509-certificates}?
       |     |  +-- client-certs?                    ts:certificates-ref
       |     |          {ts:x509-certificates}?
       |     +--:(external) {external-client-auth-supported}?
       |        +-- client-auth-defined-elsewhere?   empty
       +-- hello-params {tls-server-hello-params-config}?
       |  +---u tlscmn:hello-params-grouping
       +-- keepalives! {tls-server-keepalives}?
          +-- max-wait?       uint16
          +-- max-attempts?   uint8
```

4.2. Example Usage

This section presents two examples showing the tls-server-grouping populated with some data. These examples are effectively the same except the first configures the server identity using a local key while the second uses a key configured in a keystore. Both examples are consistent with the examples presented in Section 2 of [I-D.ietf-netconf-trust-anchors] and Section 3.2 of [I-D.ietf-netconf-keystore].

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

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

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

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

</tls-server>
```

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

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

  <!-- how this server will authenticate itself to the client -->
  <server-identity>
    <keystore-reference>
      <asymmetric-key>ex-rsa-key</asymmetric-key>
      <certificate>ex-rsa-cert</certificate>
    </keystore-reference>
  </server-identity>

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

</tls-server>
```

4.3. YANG Module

This YANG module has a normative references to [RFC5246], [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore].

```
<CODE BEGINS> file "ietf-tls-server@2019-07-02.yang"
module ietf-tls-server {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-tls-server";
  prefix tlss;

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

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

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

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

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

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

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

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

     Redistribution and use in source and binary forms, with
```

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

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

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

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

     // Features

     feature tls-server-hello-params-config {
       description
         "TLS hello message parameters are configurable on a TLS
          server.";
     }

     feature tls-server-keepalives {
       description
         "Per socket TLS keepalive parameters are configurable for
          TLS servers on the server implementing this feature.";
     }

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


     feature external-client-auth-supported {
       description
         "Indicates that the TLS server supports external
```

```
         configuration of client credentials.";
     }



     // Groupings

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

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


       container server-identity {  // FIXME: what about PSKs?
         nacm:default-deny-write;
         description
           "A locally-defined or referenced end-entity certificate,
            including any configured intermediate certificates, the
            TLS server will present when establishing a TLS connection
            in its Certificate message, as defined in Section 7.4.2
            in RFC 5246.";
         reference
           "RFC 5246:
              The Transport Layer Security (TLS) Protocol Version 1.2
            RFC ZZZZ:
              YANG Data Model for a 'Keystore' Mechanism";
         uses ks:local-or-keystore-end-entity-cert-with-key-grouping;
       } // container server-identity

       container client-authentication {  // FIXME: what about PSKs?
         nacm:default-deny-write;
         presence
           "Indicates that certificate based client authentication
            is supported (i.e., the server will request that the
            client send a certificate).";
         description
           "Specifies if TLS client authentication is required or
            optional, and specifies if the certificates needed to
```

```
            authenticate the TLS client are configured locally or
            externally.  If configured locally, the data model
            enables both trust anchors and end-entity certificate
            to be set.";
         choice required-or-optional {
           mandatory true;  // or default to 'required' ?
           description
             "Indicates if TLS-level client authentication is required
              or optional.  This is necessary for some protocols (e.g.,
              RESTCONF) the may optionally authenticate a client via
              TLS-level authentication, HTTP-level authentication, or
              both simultaneously).";
            leaf required {
              type empty;
              description
                "Indicates that TLS-level client authentication is
                 required.";
            }
            leaf optional {
              type empty;
              description
                "Indicates that TLS-level client authentication is
                 optional.";
            }
         }
         choice local-or-external {
           mandatory true;
           description
             "Indicates if the certificates needed to authenticate
              the client are configured locally or externally.  The
              need to support external configuration for client
              authentication stems from the desire to support
              consuming data models that prefer to place client
              authentication with client definitions, rather then
              in a data model principally concerned with configuring
              the transport.";
           case local {
             if-feature "local-client-auth-supported";
             description
               "The certificates needed to authenticate the clients
                are configured locally.";
             leaf ca-certs {
               if-feature "ts:x509-certificates";
               type ts:certificates-ref;//FIXME: local-or-remote?
               description
                 "A reference to a list of certificate authority (CA)
                  certificates used by the TLS server to authenticate
                  TLS client certificates.  A client certificate is
```

```
                authenticated if it has a valid chain of trust to
                a configured CA certificate.";
             reference
               "RFC YYYY: YANG Data Model for Global Trust Anchors";
           }
           leaf client-certs {
             if-feature "ts:x509-certificates";
             type ts:certificates-ref;//FIXME: local-or-remote?
             description
               "A reference to a list of client certificates
                used by the TLS server to authenticate TLS
                client certificates.  A clients certificate
                is authenticated if it is an exact match to
                a configured client certificate.";
             reference
               "RFC YYYY: YANG Data Model for Global Trust Anchors";
           }
         }
         case external {
           if-feature "external-client-auth-supported";
           description
             "The certificates needed to authenticate the clients
              are configured externally.";
           leaf client-auth-defined-elsewhere {
             type empty;
             description
               "Indicates that certificates needed to authenticate
                clients are configured elsewhere.";
           }
         }
       } // choice local-or-external
     } // container client-authentication

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

     container keepalives {
       nacm:default-deny-write;
       if-feature "tls-server-keepalives";
       presence "Indicates that keepalives are enabled.";
       description
         "Configures the keep-alive policy, to proactively test
          the aliveness of the TLS client.  An unresponsive
```

```
            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. The TLS Common Model

The TLS common model presented in this section contains identities and groupings common to both TLS clients and TLS servers. The hello-params-grouping can be used to configure the list of TLS algorithms permitted by the TLS client or TLS server. The lists of algorithms are ordered such that, if multiple algorithms are permitted by the client, the algorithm that appears first in its list that is also permitted by the server is used for the TLS transport layer connection. The ability to restrict the algorithms allowed is provided in this grouping for TLS clients and TLS servers that are capable of doing so and may serve to make TLS clients and TLS servers compliant with local security policies. This model supports both TLS1.2 [RFC5246] and TLS 1.3 [RFC8446].

TLS 1.2 and TLS 1.3 have different ways defining their own supported cryptographic algorithms, see TLS and DTLS IANA registries page (https://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml):

- o TLS 1.2 defines four categories of registries for cryptographic algorithms: TLS Cipher Suites, TLS SignatureAlgorithm, TLS HashAlgorithm, TLS Supported Groups. TLS Cipher Suites plays the role of combining all of them into one set, as each value of the set represents a unique and feasible combination of all the cryptographic algorithms, and thus the other three registry categories do not need to be considered here. In this document, the TLS common model only chooses those TLS1.2 algorithms in TLS Cipher Suites which are marked as recommended: TLS_DHE_RSA_WITH_AES_128_GCM_SHA256, TLS_DHE_RSA_WITH_AES_256_GCM_SHA384, TLS_DHE_PSK_WITH_AES_128_GCM_SHA256, TLS_DHE_PSK_WITH_AES_256_GCM_SHA384, and so on. All chosen algorithms are enumerated in Table 1-1 below;

- o TLS 1.3 defines its supported algorithms differently. Firstly, it defines three categories of registries for cryptographic algorithms: TLS Cipher Suites, TLS SignatureScheme, TLS Supported Groups. Secondly, all three of these categories are useful, since they represent different parts of all the supported algorithms respectively. Thus, all of these registries categories are considered here. In this draft, the TLS common model chooses only those TLS1.3 algorithms specified in B.4, 4.2.3, 4.2.7 of [RFC8446].

Thus, in order to support both TLS1.2 and TLS1.3, the cipher-suites part of the hello-params-grouping should include three parameters for configuring its permitted TLS algorithms, which are: TLS Cipher Suites, TLS SignatureScheme, TLS Supported Groups. Note that TLS1.2 only uses TLS Cipher Suites.

[I-D.ietf-netconf-crypto-types] defines six categories of cryptographic algorithms (hash-algorithm, symmetric-key-encryption-algorithm, mac-algorithm, asymmetric-key-encryption-algorithm, signature-algorithm, key-negotiation-algorithm) and lists several widely accepted algorithms for each of them. The TLS client and server models use one or more of these algorithms. The following tables are provided, in part to define the subset of algorithms defined in the crypto-types model used by TLS, and in part to ensure compatibility of configured TLS cryptographic parameters for configuring its permitted TLS algorithms:

ciper-suites in hello-params-grouping	HASH
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	sha-256
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	sha-384
TLS_DHE_PSK_WITH_AES_128_GCM_SHA256	sha-256
TLS_DHE_PSK_WITH_AES_256_GCM_SHA384	sha-384
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	sha-256
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	sha-384
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	sha-256
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	sha-384
TLS_DHE_RSA_WITH_AES_128_CCM	sha-256
TLS_DHE_RSA_WITH_AES_256_CCM	sha-256
TLS_DHE_PSK_WITH_AES_128_CCM	sha-256
TLS_DHE_PSK_WITH_AES_256_CCM	sha-256
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256	sha-256
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256	sha-256
TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256	sha-256
TLS_ECDHE_PSK_WITH_CHACHA20_POLY1305_SHA256	sha-256
TLS_DHE_PSK_WITH_CHACHA20_POLY1305_SHA256	sha-256
TLS_ECDHE_PSK_WITH_AES_128_GCM_SHA256	sha-256
TLS_ECDHE_PSK_WITH_AES_256_GCM_SHA384	sha-384
TLS_ECDHE_PSK_WITH_AES_128_CCM_SHA256	sha-256

Table 1-1 TLS 1.2 Compatibility Matrix Part 1: ciper-suites mapping to hash-algorithm

ciper-suites in hello-params-grouping	symmetric
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	enc-aes-128-gcm
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	enc-aes-256-gcm
TLS_DHE_PSK_WITH_AES_128_GCM_SHA256	enc-aes-128-gcm
TLS_DHE_PSK_WITH_AES_256_GCM_SHA384	enc-aes-256-gcm
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	enc-aes-128-gcm
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	enc-aes-256-gcm
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	enc-aes-128-gcm
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	enc-aes-256-gcm
TLS_DHE_RSA_WITH_AES_128_CCM	enc-aes-128-ccm
TLS_DHE_RSA_WITH_AES_256_CCM	enc-aes-256-ccm
TLS_DHE_PSK_WITH_AES_128_CCM	enc-aes-128-ccm
TLS_DHE_PSK_WITH_AES_256_CCM	enc-aes-256-ccm
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256	enc-chacha20-poly1305
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256	enc-chacha20-poly1305
TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256	enc-chacha20-poly1305
TLS_ECDHE_PSK_WITH_CHACHA20_POLY1305_SHA256	enc-chacha20-poly1305
TLS_DHE_PSK_WITH_CHACHA20_POLY1305_SHA256	enc-chacha20-poly1305
TLS_ECDHE_PSK_WITH_AES_128_GCM_SHA256	enc-aes-128-gcm
TLS_ECDHE_PSK_WITH_AES_256_GCM_SHA384	enc-aes-256-gcm
TLS_ECDHE_PSK_WITH_AES_128_CCM_SHA256	enc-aes-128-ccm

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

ciper-suites in hello-params-grouping	MAC
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	mac-aes-128-gcm
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	mac-aes-256-gcm
TLS_DHE_PSK_WITH_AES_128_GCM_SHA256	mac-aes-128-gcm
TLS_DHE_PSK_WITH_AES_256_GCM_SHA384	mac-aes-256-gcm
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	mac-aes-128-gcm
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	mac-aes-256-gcm
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	mac-aes-128-gcm
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	mac-aes-256-gcm
TLS_DHE_RSA_WITH_AES_128_CCM	mac-aes-128-ccm
TLS_DHE_RSA_WITH_AES_256_CCM	mac-aes-256-ccm
TLS_DHE_PSK_WITH_AES_128_CCM	mac-aes-128-ccm
TLS_DHE_PSK_WITH_AES_256_CCM	mac-aes-256-ccm
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256	mac-chacha20-poly1305
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256	mac-chacha20-poly1305
TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256	mac-chacha20-poly1305
TLS_ECDHE_PSK_WITH_CHACHA20_POLY1305_SHA256	mac-chacha20-poly1305
TLS_DHE_PSK_WITH_CHACHA20_POLY1305_SHA256	mac-chacha20-poly1305
TLS_ECDHE_PSK_WITH_AES_128_GCM_SHA256	mac-aes-128-gcm
TLS_ECDHE_PSK_WITH_AES_256_GCM_SHA384	mac-aes-256-gcm
TLS_ECDHE_PSK_WITH_AES_128_CCM_SHA256	mac-aes-128-ccm

Table 1-3 TLS 1.2 Compatibility Matrix Part 3: ciper-suites mapping to MAC-algorithm

ciper-suites in hello-params-grouping	signature
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	rsa-pkcs1-sha256
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	rsa-pkcs1-sha384
TLS_DHE_PSK_WITH_AES_128_GCM_SHA256	N/A
TLS_DHE_PSK_WITH_AES_256_GCM_SHA384	N/A
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	ecdsa-secp256r1-sha256
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	ecdsa-secp384r1-sha384
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	rsa-pkcs1-sha256
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	rsa-pkcs1-sha384
TLS_DHE_RSA_WITH_AES_128_CCM	rsa-pkcs1-sha256
TLS_DHE_RSA_WITH_AES_256_CCM	rsa-pkcs1-sha256
TLS_DHE_PSK_WITH_AES_128_CCM	N/A
TLS_DHE_PSK_WITH_AES_256_CCM	N/A
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256	rsa-pkcs1-sha256
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256	ecdsa-secp256r1-sha256
TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256	rsa-pkcs1-sha256
TLS_ECDHE_PSK_WITH_CHACHA20_POLY1305_SHA256	N/A
TLS_DHE_PSK_WITH_CHACHA20_POLY1305_SHA256	N/A
TLS_ECDHE_PSK_WITH_AES_128_GCM_SHA256	N/A
TLS_ECDHE_PSK_WITH_AES_256_GCM_SHA384	N/A
TLS_ECDHE_PSK_WITH_AES_128_CCM_SHA256	N/A

Table 1-4 TLS 1.2 Compatibility Matrix Part 4: ciper-suites mapping to signature-algorithm

ciper-suites in hello-params-grouping	key-negotiation
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	dhe-ffdhe2048, ...
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	dhe-ffdhe2048, ...
TLS_DHE_PSK_WITH_AES_128_GCM_SHA256	psk-dhe-ffdhe2048, ...
TLS_DHE_PSK_WITH_AES_256_GCM_SHA384	psk-dhe-ffdhe2048, ...
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	ecdhe-secp256r1, ...
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	ecdhe-secp256r1, ...
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	ecdhe-secp256r1, ...
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ecdhe-secp256r1, ...
TLS_DHE_RSA_WITH_AES_128_CCM	dhe-ffdhe2048, ...
TLS_DHE_RSA_WITH_AES_256_CCM	dhe-ffdhe2048, ...
TLS_DHE_PSK_WITH_AES_128_CCM	psk-dhe-ffdhe2048, ...
TLS_DHE_PSK_WITH_AES_256_CCM	psk-dhe-ffdhe2048, ...
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256	ecdhe-secp256r1, ...
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256	ecdhe-secp256r1, ...
TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256	dhe-ffdhe2048, ...
TLS_ECDHE_PSK_WITH_CHACHA20_POLY1305_SHA256	psk-ecdhe-secp256r1,...
TLS_DHE_PSK_WITH_CHACHA20_POLY1305_SHA256	psk-dhe-ffdhe2048, ...
TLS_ECDHE_PSK_WITH_AES_128_GCM_SHA256	psk-ecdhe-secp256r1,...
TLS_ECDHE_PSK_WITH_AES_256_GCM_SHA384	psk-ecdhe-secp256r1,...
TLS_ECDHE_PSK_WITH_AES_128_CCM_SHA256	psk-ecdhe-secp256r1,...

Table 1-5 TLS 1.2 Compatibility Matrix Part 5: ciper-suites mapping to key-negotiation-algorithm

ciper-suites in hello -params-grouping	HASH
TLS_AES_128_GCM_SHA256	sha-256
TLS_AES_256_GCM_SHA384	sha-384
TLS_CHACHA20_POLY1305_SHA256	sha-256
TLS_AES_128_CCM_SHA256	sha-256

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

ciper-suites in hello -params-grouping	symmetric
TLS_AES_128_GCM_SHA256	enc-aes-128-gcm
TLS_AES_256_GCM_SHA384	enc-aes-128-gcm
TLS_CHACHA20_POLY1305_SHA256	enc-chacha20-poly1305
TLS_AES_128_CCM_SHA256	enc-aes-128-ccm

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

ciper-suites in hello -params-grouping	symmetric
TLS_AES_128_GCM_SHA256	mac-aes-128-gcm
TLS_AES_256_GCM_SHA384	mac-aes-128-gcm
TLS_CHACHA20_POLY1305_SHA256	mac-chacha20-poly1305
TLS_AES_128_CCM_SHA256	mac-aes-128-ccm

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

signatureScheme in hello -params-grouping	signature
rsa-pkcs1-sha256	rsa-pkcs1-sha256
rsa-pkcs1-sha384	rsa-pkcs1-sha384
rsa-pkcs1-sha512	rsa-pkcs1-sha512
rsa-pss-rsae-sha256	rsa-pss-rsae-sha256
rsa-pss-rsae-sha384	rsa-pss-rsae-sha384
rsa-pss-rsae-sha512	rsa-pss-rsae-sha512
rsa-pss-pss-sha256	rsa-pss-pss-sha256
rsa-pss-pss-sha384	rsa-pss-pss-sha384
rsa-pss-pss-sha512	rsa-pss-pss-sha512
ecdsa-secp256r1-sha256	ecdsa-secp256r1-sha256
ecdsa-secp384r1-sha384	ecdsa-secp384r1-sha384
ecdsa-secp521r1-sha512	ecdsa-secp521r1-sha512
ed25519	ed25519
ed448	ed448

Table 2-4 TLS 1.3 Compatibility Matrix Part 4: SignatureScheme mapping to signature-algorithm

supported Groups in hello -params-grouping	key-negotiation
dhe-ffdhe2048	dhe-ffdhe2048
dhe-ffdhe3072	dhe-ffdhe3072
dhe-ffdhe4096	dhe-ffdhe4096
dhe-ffdhe6144	dhe-ffdhe6144
dhe-ffdhe8192	dhe-ffdhe8192
psk-dhe-ffdhe2048	psk-dhe-ffdhe2048
psk-dhe-ffdhe3072	psk-dhe-ffdhe3072
psk-dhe-ffdhe4096	psk-dhe-ffdhe4096
psk-dhe-ffdhe6144	psk-dhe-ffdhe6144
psk-dhe-ffdhe8192	psk-dhe-ffdhe8192
ecdhe-secp256r1	ecdhe-secp256r1
ecdhe-secp384r1	ecdhe-secp384r1
ecdhe-secp521r1	ecdhe-secp521r1
ecdhe-x25519	ecdhe-x25519
ecdhe-x448	ecdhe-x448
psk-ecdhe-secp256r1	psk-ecdhe-secp256r1
psk-ecdhe-secp384r1	psk-ecdhe-secp384r1
psk-ecdhe-secp521r1	psk-ecdhe-secp521r1
psk-ecdhe-x25519	psk-ecdhe-x25519
psk-ecdhe-x448	psk-ecdhe-x448

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

Note that in Table 1-5:

- o dhe-ffdhe2048, ... is the abbreviation of dhe-ffdhe2048, dhe-ffdhe3072, dhe-ffdhe4096, dhe-ffdhe6144, dhe-ffdhe8192;

- o psk-dhe-ffdhe2048, ... is the abbreviation of psk-dhe-ffdhe2048, psk-dhe-ffdhe3072, psk-dhe-ffdhe4096, psk-dhe-ffdhe6144, psk-dhe-ffdhe8192;

- o ecdhe-secp256r1, ... is the abbreviation of ecdhe-secp256r1, ecdhe-secp384r1, ecdhe-secp521r1, ecdhe-x25519, ecdhe-x448;

- o psk-ecdhe-secp256r1, ... is the abbreviation of psk-ecdhe-secp256r1, psk-ecdhe-secp384r1, psk-ecdhe-secp521r1, psk-ecdhe-x25519, psk-ecdhe-x448.

Features are defined for algorithms that are OPTIONAL or are not widely supported by popular implementations. Note that the list of algorithms is not exhaustive.

5.1. Tree Diagram

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

```
module: ietf-tls-common

  grouping hello-params-grouping
    +-- tls-versions
    |  +-- tls-version*   identityref
    +-- cipher-suites
       +-- cipher-suite*   identityref
```

5.2. Example Usage

This section shows how it would appear if the transport-params-grouping were populated with some data.

```
<hello-params
   xmlns="urn:ietf:params:xml:ns:yang:ietf-tls-common"
   xmlns:tlscmn="urn:ietf:params:xml:ns:yang:ietf-tls-common">
  <tls-versions>
    <tls-version>tlscmn:tls-1.1</tls-version>
    <tls-version>tlscmn:tls-1.2</tls-version>
  </tls-versions>
  <cipher-suites>
    <cipher-suite>tlscmn:dhe-rsa-with-aes-128-cbc-sha</cipher-suite>
    <cipher-suite>tlscmn:rsa-with-aes-128-cbc-sha</cipher-suite>
    <cipher-suite>tlscmn:rsa-with-3des-ede-cbc-sha</cipher-suite>
  </cipher-suites>
</hello-params>
```

5.3. YANG Module

This YANG module has a normative references to [RFC4346], [RFC5246], [RFC5288], [RFC5289], and [RFC8422].

This YANG module has a informative references to [RFC2246], [RFC4346], [RFC5246], and [RFC8446].

```
<CODE BEGINS> file "ietf-tls-common@2019-07-02.yang"
module ietf-tls-common {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-tls-common";
  prefix tlscmn;

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

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

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

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

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

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

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

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

     // Features

     feature tls-1_0 {
       description
         "TLS Protocol Version 1.0 is supported.";
       reference
         "RFC 2246: The TLS Protocol Version 1.0";
     }

     feature tls-1_1 {
```

```
        description
          "TLS Protocol Version 1.1 is supported.";
        reference
          "RFC 4346: The Transport Layer Security (TLS) Protocol
                     Version 1.1";
      }

      feature tls-1_2 {
        description
          "TLS Protocol Version 1.2 is supported.";
        reference
          "RFC 5246: The Transport Layer Security (TLS) Protocol
                     Version 1.2";
      }

      feature tls-1_3 {
        description
          "TLS Protocol Version 1.2 is supported.";
        reference
          "RFC 8446: The Transport Layer Security (TLS) Protocol
                     Version 1.3";
      }

      feature tls-ecc {
        description
          "Elliptic Curve Cryptography (ECC) is supported for TLS.";
        reference
          "RFC 8422: Elliptic Curve Cryptography (ECC) Cipher Suites
                     for Transport Layer Security (TLS)";
      }

      feature tls-dhe {
        description
          "Ephemeral Diffie-Hellman key exchange is supported for TLS.";
        reference
          "RFC 5246: The Transport Layer Security (TLS) Protocol
                     Version 1.2";
      }

      feature tls-3des {
        description
          "The Triple-DES block cipher is supported for TLS.";
        reference
          "RFC 5246: The Transport Layer Security (TLS) Protocol
                     Version 1.2";
      }

      feature tls-gcm {
```

```
         description
           "The Galois/Counter Mode authenticated encryption mode is
            supported for TLS.";
         reference
           "RFC 5288: AES Galois Counter Mode (GCM) Cipher Suites for
                      TLS";
       }

       feature tls-sha2 {
         description
           "The SHA2 family of cryptographic hash functions is supported
            for TLS.";
         reference
           "FIPS PUB 180-4: Secure Hash Standard (SHS)";
       }

       // Identities

       identity tls-version-base {
         description
           "Base identity used to identify TLS protocol versions.";
       }

       identity tls-1.0 {
         base tls-version-base;
         if-feature "tls-1_0";
         description
           "TLS Protocol Version 1.0.";
         reference
           "RFC 2246: The TLS Protocol Version 1.0";
       }

       identity tls-1.1 {
         base tls-version-base;
         if-feature "tls-1_1";
         description
           "TLS Protocol Version 1.1.";
         reference
           "RFC 4346: The Transport Layer Security (TLS) Protocol
                      Version 1.1";
       }

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

```
         "RFC 5246: The Transport Layer Security (TLS) Protocol
                    Version 1.2";
     }

     identity cipher-suite-base {
       description
         "Base identity used to identify TLS cipher suites.";
     }

     identity rsa-with-aes-128-cbc-sha {
       base cipher-suite-base;
       description
         "Cipher suite TLS_RSA_WITH_AES_128_CBC_SHA.";
       reference
         "RFC 5246: The Transport Layer Security (TLS) Protocol
                    Version 1.2";
     }

     identity rsa-with-aes-256-cbc-sha {
       base cipher-suite-base;
       description
         "Cipher suite TLS_RSA_WITH_AES_256_CBC_SHA.";
       reference
         "RFC 5246: The Transport Layer Security (TLS) Protocol
                    Version 1.2";
     }

     identity rsa-with-aes-128-cbc-sha256 {
       base cipher-suite-base;
       if-feature "tls-sha2";
       description
         "Cipher suite TLS_RSA_WITH_AES_128_CBC_SHA256.";
       reference
         "RFC 5246: The Transport Layer Security (TLS) Protocol
                    Version 1.2";
     }

     identity rsa-with-aes-256-cbc-sha256 {
       base cipher-suite-base;
       if-feature "tls-sha2";
       description
         "Cipher suite TLS_RSA_WITH_AES_256_CBC_SHA256.";
       reference
         "RFC 5246: The Transport Layer Security (TLS) Protocol
                    Version 1.2";
     }

     identity dhe-rsa-with-aes-128-cbc-sha {
```

```
       base cipher-suite-base;
       if-feature "tls-dhe";
       description
         "Cipher suite TLS_DHE_RSA_WITH_AES_128_CBC_SHA.";
       reference
         "RFC 5246: The Transport Layer Security (TLS) Protocol
                    Version 1.2";
     }

     identity dhe-rsa-with-aes-256-cbc-sha {
       base cipher-suite-base;
       if-feature "tls-dhe";
       description
         "Cipher suite TLS_DHE_RSA_WITH_AES_256_CBC_SHA.";
       reference
         "RFC 5246: The Transport Layer Security (TLS) Protocol
                    Version 1.2";
     }

     identity dhe-rsa-with-aes-128-cbc-sha256 {
       base cipher-suite-base;
       if-feature "tls-dhe and tls-sha2";
       description
         "Cipher suite TLS_DHE_RSA_WITH_AES_128_CBC_SHA256.";
       reference
         "RFC 5246: The Transport Layer Security (TLS) Protocol
                    Version 1.2";
     }

     identity dhe-rsa-with-aes-256-cbc-sha256 {
       base cipher-suite-base;
       if-feature "tls-dhe and tls-sha2";
       description
         "Cipher suite TLS_DHE_RSA_WITH_AES_256_CBC_SHA256.";
       reference
         "RFC 5246: The Transport Layer Security (TLS) Protocol
                    Version 1.2";
     }

     identity ecdhe-ecdsa-with-aes-128-cbc-sha256 {
       base cipher-suite-base;
       if-feature "tls-ecc and tls-sha2";
       description
         "Cipher suite TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256.";
       reference
         "RFC 5289: TLS Elliptic Curve Cipher Suites with
                    SHA-256/384 and AES Galois Counter Mode (GCM)";
     }
```

```
     identity ecdhe-ecdsa-with-aes-256-cbc-sha384 {
       base cipher-suite-base;
       if-feature "tls-ecc and tls-sha2";
       description
         "Cipher suite TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384.";
       reference
         "RFC 5289: TLS Elliptic Curve Cipher Suites with
                    SHA-256/384 and AES Galois Counter Mode (GCM)";
     }

     identity ecdhe-rsa-with-aes-128-cbc-sha256 {
       base cipher-suite-base;
       if-feature "tls-ecc and tls-sha2";
       description
         "Cipher suite TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256.";
       reference
         "RFC 5289: TLS Elliptic Curve Cipher Suites with
                    SHA-256/384 and AES Galois Counter Mode (GCM)";
     }

     identity ecdhe-rsa-with-aes-256-cbc-sha384 {
       base cipher-suite-base;
       if-feature "tls-ecc and tls-sha2";
       description
         "Cipher suite TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384.";
       reference
         "RFC 5289: TLS Elliptic Curve Cipher Suites with
                    SHA-256/384 and AES Galois Counter Mode (GCM)";
     }

     identity ecdhe-ecdsa-with-aes-128-gcm-sha256 {
       base cipher-suite-base;
       if-feature "tls-ecc and tls-gcm and tls-sha2";
       description
         "Cipher suite TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256.";
       reference
         "RFC 5289: TLS Elliptic Curve Cipher Suites with
                    SHA-256/384 and AES Galois Counter Mode (GCM)";
     }

     identity ecdhe-ecdsa-with-aes-256-gcm-sha384 {
       base cipher-suite-base;
       if-feature "tls-ecc and tls-gcm and tls-sha2";
       description
         "Cipher suite TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384.";
       reference
         "RFC 5289: TLS Elliptic Curve Cipher Suites with
                    SHA-256/384 and AES Galois Counter Mode (GCM)";
```

```
     }

     identity ecdhe-rsa-with-aes-128-gcm-sha256 {
       base cipher-suite-base;
       if-feature "tls-ecc and tls-gcm and tls-sha2";
       description
         "Cipher suite TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256.";
       reference
         "RFC 5289: TLS Elliptic Curve Cipher Suites with
                    SHA-256/384 and AES Galois Counter Mode (GCM)";
     }

     identity ecdhe-rsa-with-aes-256-gcm-sha384 {
       base cipher-suite-base;
       if-feature "tls-ecc and tls-gcm and tls-sha2";
       description
         "Cipher suite TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384.";
       reference
         "RFC 5289: TLS Elliptic Curve Cipher Suites with
                    SHA-256/384 and AES Galois Counter Mode (GCM)";
     }

     identity rsa-with-3des-ede-cbc-sha {
       base cipher-suite-base;
       if-feature "tls-3des";
       description
         "Cipher suite TLS_RSA_WITH_3DES_EDE_CBC_SHA.";
       reference
         "RFC 5246: The Transport Layer Security (TLS) Protocol
                    Version 1.2";
     }

     identity ecdhe-rsa-with-3des-ede-cbc-sha {
       base cipher-suite-base;
       if-feature "tls-ecc and tls-3des";
       description
         "Cipher suite TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA.";
       reference
         "RFC 8422: Elliptic Curve Cryptography (ECC) Cipher Suites
                    for Transport Layer Security (TLS)";
     }

     identity ecdhe-rsa-with-aes-128-cbc-sha {
       base cipher-suite-base;
       if-feature "tls-ecc";
       description
         "Cipher suite TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA.";
       reference
```

```
         "RFC 8422: Elliptic Curve Cryptography (ECC) Cipher Suites
                    for Transport Layer Security (TLS)";
     }

     identity ecdhe-rsa-with-aes-256-cbc-sha {
       base cipher-suite-base;
       if-feature "tls-ecc";
       description
         "Cipher suite TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA.";
       reference
         "RFC 8422: Elliptic Curve Cryptography (ECC) Cipher Suites
                    for Transport Layer Security (TLS)";
     }

     // Groupings

     grouping hello-params-grouping {
       description
         "A reusable grouping for TLS hello message parameters.";
       reference
         "RFC 5246: The Transport Layer Security (TLS) Protocol
                    Version 1.2";
       container tls-versions {
         description
           "Parameters regarding TLS versions.";
         leaf-list tls-version {
           type identityref {
             base tls-version-base;
           }
           description
             "Acceptable TLS protocol versions.

              If this leaf-list is not configured (has zero elements)
              the acceptable TLS protocol versions are implementation-
              defined.";
         }
       }
       container cipher-suites {
         description
           "Parameters regarding cipher suites.";
         leaf-list cipher-suite {
           type identityref {
             base cipher-suite-base;
           }
           ordered-by user;
           description
             "Acceptable cipher suites in order of descending
              preference.  The configured host key algorithms should
```

```
              be compatible with the algorithm used by the configured
              private key.  Please see Section 5 of RFC XXXX for
              valid combinations.

              If this leaf-list is not configured (has zero elements)
              the acceptable cipher suites are implementation-
              defined.";
           reference
             "RFC XXXX: YANG Groupings for TLS Clients and TLS Servers";
         }
       }
     }
   }
   <CODE ENDS>
```

6. Security Considerations

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

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

Since the modules in this document only define groupings, these considerations are primarily for the designers of other modules that use these groupings.

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

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

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

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

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

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

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

7. IANA Considerations

7.1. The IETF XML Registry

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

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

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

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

7.2. The YANG Module Names Registry

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

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

name:         ietf-tls-server
namespace:    urn:ietf:params:xml:ns:yang:ietf-tls-server
prefix:       tlss
reference:    RFC XXXX

name:         ietf-tls-common
namespace:    urn:ietf:params:xml:ns:yang:ietf-tls-common
prefix:       tlscmn
reference:    RFC XXXX
```

8. References

8.1. Normative References

[I-D.ietf-netconf-crypto-types]
Watsen, K. and H. Wang, "Common YANG Data Types for Cryptography", draft-ietf-netconf-crypto-types-09 (work in progress), June 2019.

[I-D.ietf-netconf-keystore]
Watsen, K., "A YANG Data Model for a Keystore", draft-ietf-netconf-keystore-11 (work in progress), June 2019.

[I-D.ietf-netconf-trust-anchors]
Watsen, K., "A YANG Data Model for a Truststore", draft-ietf-netconf-trust-anchors-05 (work in progress), June 2019.

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

[RFC5288] Salowey, J., Choudhury, A., and D. McGrew, "AES Galois Counter Mode (GCM) Cipher Suites for TLS", RFC 5288, DOI 10.17487/RFC5288, August 2008, <https://www.rfc-editor.org/info/rfc5288>.

[RFC5289] Rescorla, E., "TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM)", RFC 5289, DOI 10.17487/RFC5289, August 2008, <https://www.rfc-editor.org/info/rfc5289>.

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

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

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

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

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

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

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

8.2. Informative References

[RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, DOI 10.17487/RFC2246, January 1999, <https://www.rfc-editor.org/info/rfc2246>.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <https://www.rfc-editor.org/info/rfc2818>.

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

[RFC4346] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, DOI 10.17487/RFC4346, April 2006, <https://www.rfc-editor.org/info/rfc4346>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

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

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

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

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

Appendix A. Change Log

A.1. 00 to 01

- Noted that '0.0.0.0' and '::' might have special meanings.

- Renamed "keychain" to "keystore".

A.2. 01 to 02

- Removed the groupings containing transport-level configuration. Now modules contain only the transport-independent groupings.

- Filled in previously incomplete 'ietf-tls-client' module.

- Added cipher suites for various algorithms into new 'ietf-tls-common' module.

A.3. 02 to 03

- Added a 'must' statement to container 'server-auth' asserting that at least one of the various auth mechanisms must be specified.

- Fixed description statement for leaf 'trusted-ca-certs'.

A.4. 03 to 04

- Updated title to "YANG Groupings for TLS Clients and TLS Servers"

- Updated leafref paths to point to new keystore path

- Changed the YANG prefix for ietf-tls-common from 'tlscom' to 'tlscmn'.

- Added TLS protocol verions 1.0 and 1.1.

- Made author lists consistent

- Now tree diagrams reference ietf-netmod-yang-tree-diagrams

- Updated YANG to use typedefs around leafrefs to common keystore paths

- Now inlines key and certificates (no longer a leafref to keystore)

A.5. 04 to 05

- Merged changes from co-author.

A.6. 05 to 06

- Updated to use trust anchors from trust-anchors draft (was keystore draft)

- Now Uses new keystore grouping enabling asymmetric key to be either locally defined or a reference to the keystore.

A.7. 06 to 07

- factored the tls-[client|server]-groupings into more reusable groupings.

- added if-feature statements for the new "x509-certificates" feature defined in draft-ietf-netconf-trust-anchors.

A.8. 07 to 08

- Added a number of compatibility matrices to Section 5 (thanks Frank!)

- Clarified that any configured "cipher-suite" values need to be compatible with the configured private key.

A.9. 08 to 09

- Updated examples to reflect update to groupings defined in the keystore draft.

- Add TLS keepalives features and groupings.

- Prefixed top-level TLS grouping nodes with 'tls-' and support mashups.

- Updated copyright date, boilerplate template, affiliation, and folding algorithm.

A.10. 09 to 10

- Reformatted the YANG modules.

A.11. 10 to 11

- Collapsed all the inner groupings into the top-level grouping.

- Added a top-level "demux container" inside the top-level grouping.

- Added NACM statements and updated the Security Considerations section.

- Added "presence" statements on the "keepalive" containers, as was needed to address a validation error that appeared after adding the "must" statements into the NETCONF/RESTCONF client/server modules.

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

A.12. 11 to 12

- In server model, made 'client-authentication' a 'presence' node indicating that the server supports client authentication.

- In the server model, added a 'required-or-optional' choice to 'client-authentication' to better support protocols such as RESTCONF.

- In the server model, added a 'local-or-external' choice to 'client-authentication' to better support consuming data models that prefer to keep client auth with client definitions than in a model principally concerned with the "transport".

- In both models, removed the "demux containers", floating the nacm:default-deny-write to each descendent node, and adding a note to model designers regarding the potential need to add their own demux containers.

- Fixed a couple references (section 2 --> section 3)

A.13. 12 to 13

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

A.14. 12 to 13

- Removed 'container' under 'client-identity' to match server model.

- Updated examples to reflect change grouping in keystore module.

A.15. 13 to 14

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

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

Acknowledgements

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

Authors' Addresses

Kent Watsen
Watsen Networks

EMail: kent+ietf@watsen.net

Gary Wu
Cisco Systems

EMail: garywu@cisco.com

Liang Xia
Huawei

EMail: frank.xialiang@huawei.com

NETCONF G. Zheng
Internet-Draft T. Zhou
Intended status: Standards Track A. Clemm
Expires: September 12, 2019 Huawei
March 11, 2019

UDP based Publication Channel for Streaming Telemetry
draft-ietf-netconf-udp-pub-channel-05

Abstract

This document describes a UDP-based publication channel for streaming telemetry use to collect data from devices. A new shim header is proposed to facilitate the distributed data collection mechanism which directly pushes data from line cards to the collector. Because of the lightweight UDP encapsulation, higher frequency and better transit performance can be achieved.

Requirements Language

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

Table of Contents

1. Introduction

Streaming telemetry refers to sending a continuous stream of operational data from a device to a remote receiver. This provides an ability to monitor a network from remote and to provide network analytics. Devices generate telemetry data and push that data to a collector for further analysis. By streaming the data, much better performance, finer-grained sampling, monitoring accuracy, and bandwidth utilization can be achieved than with polling-based alternatives.

Sub-Notif [I-D.ietf-netconf-subscribed-notifications] defines a mechanism that allows a collector to subscribe to updates of YANG-defined data that is maintained in a YANG [RFC7950] datastore. The mechanism separates the management and control of subscriptions from the transport that is used to actually stream and deliver the data. Two transports, NETCONF transport [I-D.ietf-netconf-netconf-event-notifications] and HTTP transport [I-D.ietf-netconf-restconf-notif], have been defined so far for the notification messages.

While powerful in its features and general in its architecture, in its current form the mechanism needs to be extended to stream telemetry data at high velocity from devices that feature a distributed architecture. The transports that have been defined so far, NETCONF and HTTP, are ultimately based on TCP and lack the efficiency needed to stream data continuously at high velocity. A lighter-weight, more efficient transport, e.g. a transport based on UDP is needed.

o Firstly, data collector will suffer a lot of TCP connections from, for example, many line cards equipped on different devices.

o Secondly, as no connection state needs to be maintained, UDP encapsulation can be easily implemented by hardware which will further improve the performance.

o Thirdly, because of the lightweight UDP encapsulation, higher frequency and better transit performance can be achieved, which is important for streaming telemetry.

This document specifies a higher-performance transport option for Sub-Notif that leverages UDP. Specifically, it facilitates the distributed data collection mechanism described in [I-D.zhou-netconf-multi-stream-originators]. In the case of data originating from multiple line cards, the centralized design requires data to be internally forwarded from those line cards to the push server, presumably on a main board, which then combines the individual data items into a single consolidated stream. The centralized data collection mechanism can result in a performance bottleneck, especially when large amounts of data are involved. What is needed instead is the support for a distributed mechanism that allows to directly push multiple individual substreams, e.g. one from each line card, without needing to first pass them through an additional processing stage for internal consolidation, but still allowing those substreams to be managed and controlled via a single subscription. The proposed UDP based Publication Channel (UPC) natively supports the distributed data collection mechanism.

The transport described in this document can be used for transmitting notification messages over both IPv4 and IPv6 [RFC8200].

While this document will focus on the data publication channel, the subscription can be used in conjunction with the mechanism proposed in [I-D.ietf-netconf-subscribed-notifications] with extensions [I-D.zhou-netconf-multi-stream-originators].

2. Terminologies

Streaming Telemetry: refers to sending a continuous stream of operational data from a device to a remote receiver. This provides an ability to monitor a network from remote and to provide network analytics.

Component Subscription: A subscription that defines the data from each individual telemetry source which is managed and controlled by a single Subscription Server.

Component Subscription Server: An agent that streams telemetry data per the terms of a component subscription.

3. Transport Mechanisms

For a complete pub-sub mechanism, this section will describe how the UPC is used to interact with the Subscription Channel relying on NETCONF or RESTCONF.

3.1. Dynamic Subscription

Dynamic subscriptions for Sub-Notif are configured and managed via signaling messages transported over NETCONF [RFC6241] or RESTCONF [RFC8040]. The Sub-Notif defined RPCs which are sent and responded via the Subscription Channel (a), between the Subscriber and the Subscription Server of the Publisher. In this case, only one Receiver is associated with the Subscriber. In the Publisher, there may be multiple data originators. Notification messages are pushed on separate channels (b), from different data originators to the Receiver.

```
    +--------------+                         +--------------+
    |  Collector   |                         |  Publisher   |
    |              |                         |              |
    |  (a)   (b)   |                         |  (a)    (b)  |
    +--+------+----+                         +--+-------+---+
       |      |                                 |       |
       |      |     RPC:establish-subscription  |       |
       +---------------------------------------->       |
       |      |     RPC Reply: OK               |       |
       <----------------------------------------+       |
       |      |     UPC:notifications           |       |
       |      <-----------------------------------------+
       |      |                                 |       |
       |      |     RPC:modify-subscription     |       |
       +---------------------------------------->       |
       |      |     RPC Reply: OK               |       |
       <----------------------------------------+       |
       |      |     UPC:notifications           |       |
       |      <-----------------------------------------+
       |      |                                 |       |
       |      |     RPC:delete-subscription     |       |
       +---------------------------------------->       |
       |      |     RPC Reply: OK               |       |
       <----------------------------------------+       |
       |      |                                 |       |
       |      |                                 |       |
       +      +                                 +       +
```

Fig. 2 Call Flow For Dynamic Subscription

In the case of dynamic subscription, the Receiver and the Subscriber SHOULD be colocated. So UPC can use the source IP address of the Subscription Channel as it's destination IP address. The Receiver MUST support listening messages at the IANA-assigned PORT-X or PORT-Y, but MAY be configured to listen at a different port.

For dynamic subscription, the Publication Channels MUST share fate with the subscription session. In other words, when the delete-subscription is received or the subscription session is broken, all the associated Publication Channels MUST be closed.

3.2. Configured Subscription

For a Configured Subscription, there is no guarantee that the Subscriber is currently in place with the associated Receiver(s). As defined in Sub-Notif, the subscription configuration contains the location information of all the receivers, including the IP address and the port number. So that the data originator can actively send generated messages to the corresponding Receivers via the UPC.

The first message MUST be a separate subscription-started notification to indicate the Receiver that the pushing is started. Then, the notifications can be sent immediately without any wait.

All the subscription state notifications, as defined in [I-D.ietf-netconf-subscribed-notifications], MUST be encapsulated to be separated notification messages.

```
   +---------------+                        +---------------+
   |  Collector    |                        |  Publisher    |
   |               |                        |               |
   |  (a)   (b)    |                        |  (a)    (b)   |
   +--+------+-----+                        +--+-------+----+
      |      |                                 |       |
      |      |     Capability Exchange         |       |
      <---------------------------------------->       |
      |      |                                 |       |
      |      |     Edit config(create)         |       |
      +---------------------------------------->       |
      |      |     RPC Reply: OK               |       |
      <----------------------------------------+       |
      |      |     UPC:subscription started    |       |
      |      <-----------------------------------------+
      |      |     UPC:notifications           |       |
      |      <-----------------------------------------+
      |      |                                 |       |
      |      |     Edit config(delete)         |       |
      +---------------------------------------->       |
      |      |     RPC Reply: OK               |       |
      <----------------------------------------+       |
      |      |     UPC:subscription terminated |       |
      |      <-----------------------------------------+
      |      |                                 |       |
      |      |                                 |       |
      +      +                                 +       +
```

Fig. 3 Call Flow For Configured Subscription

4. UDP Transport for Publication Channel

4.1. Design Overview

As specified in Sub-Notif, the telemetry data is encapsulated in the NETCONF/RESTCONF notification message, which is then encapsulated and carried in the transport protocols, e.g. TLS, HTTP2. The following figure shows the overview of the typical UPC message structure.

- o The Message Header contains information that can facilitate the message transmission before de-serializing the notification message.

- o Notification Message is the encoded content that the publication channel transports. The common encoding method includes GPB [1], CBOR [RFC7049], JSON, and XML. [I-D.ietf-netconf-notification-messages] describes the structure of the Notification Message for both single notification and multiple bundled notifications.

```
+-------+  +---------------+  +---------------+
|  UDP  |  |    Message    |  | Notification  |
|       |  |    Header     |  | Message       |
+-------+  +---------------+  +---------------+
```

Fig. 4 UDP Publication Message Overview

4.2. Data Format of the UPC Message Header

The UPC Message Header contains information that can facilitate the message transmission before de-serializing the notification message. The data format is shown as follows.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-------+---------------+-------+-------------------------------+
| Vers. |     Flag      |  ET   |        Length                 |
+-------+---------------+-------+-------------------------------+
|                      Message-Generator-ID                     |
+---------------------------------------------------------------+
|                      Message ID                               |
+---------------------------------------------------------------+
~                      Options                                  ~
+---------------------------------------------------------------+
```

Fig. 3 UPC Message Header Format

The Message Header contains the following field:

- o Vers.: represents the PDU (Protocol Data Unit) encoding version. The initial version value is 0.

- o Flag: is a bitmap indicating what features this packet has and the corresponding options attached. Each bit associates to one feature and one option data. When the bit is set to 1, the associated feature is enabled and the option data is attached. The sequence of the presence of the options follows the bit order of the bitmap. In this document, the flag is specified as follows:

  * bit 0, the reliability flag;

  * bit 1, the fragmentation flag;

  * other bits are reserved.

- o ET: is a 4 bits identifier to indicate the encoding type used for the Notification Message. 16 types of encoding can be expressed:

  * 0: GPB;

  * 1: CBOR;

  * 2: JSON;

  * 3: XML;

  * others are reserved.

- o Length: is the total length of the message, measured in octets, including message header.

- o Message-Generator-ID: is a 32-bit identifier of the process which created the notification message. This allows disambiguation of an information source, such as the identification of different line cards sending the notification messages. The source IP address of the UDP datagrams SHOULD NOT be interpreted as the identifier for the host that originated the UPC message. The entity sending the UPC message could be merely a relay.

- o The Message ID is generated continuously by the message generator. Different subscribers share the same notification ID sequence.

- o Options: is a variable-length field. The details of the Options will be described in the respective sections below.

4.3. Options

The order of packing the data fields in the Options field follows the bit order of the Flag field.

4.3.1. Reliability Option

The UDP based publication transport described in this document provides two streaming modes, the reliable mode an the unreliable mode, for different SLA (Service Level Agreement) and telemetry requirements.

In the unreliable streaming mode, the line card pushes the encapsulated data to the data collector without any sequence information. So the subscriber does not know whether the data is correctly received or not. Hence no retransmission happens.

The reliable streaming mode provides sequence information in the UDP packet, based on which the subscriber can deduce the packet loss and disorder. Then the subscriber can decide whether to request the retransmission of the lost packets.

In most case, the unreliable streaming mode is preferred. Because the reliable streaming mode will cost more network bandwidth and precious device resource. Different from the unreliable streaming mode, the line card cannot remove the sent reliable notifications immediately, but to keep them in the memory for a while. Reliable notifications may be pushed multiple times, which will increase the traffic. When choosing the reliable streaming mode or the unreliable streaming mode, the operate need to consider the reliable requirement together with the resource usage.

When the reliability flag bit is set to 1 in the Flag field, the following option data will be attached

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------------------------------------------------------+
|            Previous Message ID                                |
+---------------------------------------------------------------+
```

Fig. 4 Reliability Option Format

Current Message ID and Previous Message ID will be added in the packets.

For example, there are two subscriber A and B,

- o Message IDs for the generator are : [1, 2, 3, 4, 5, 6, 7, 8, 9], in which Subscriber A subscribes [1, 2, 3, 6, 7] and Subscriber B subscribes [1, 2, 4, 5, 7, 8, 9].

- o Subscriber A will receive [Previous Message ID, Current Message ID] like: [0, 1] [1, 2] [2, 3] [3, 6] [6, 7].

- o Subscriber B will receive [Previous Message ID, Current Message ID] like: [0, 1] [1, 2] [2, 4] [4, 5] [5, 7] [7, 8] [8, 9].

4.3.2. Fragmentation Option

UDP palyload has a theoretical length limitation to 65535. Other encapsulation headers will make the actual payload even shorter. Binary encodings like GPB and CBOR can make the message compact. So that the message can be encapsulated within one UDP packet, hence fragmentation will not easily happen. However, text encodings like JSON and XML can easily make the message exceed the UDP length limitation.

The Fragmentation Option can help not Application layer can split the YANG tree into several leaves. Or table into several rows. But the leaf or the row cannot be split any further. Now we consider a very long path. Since the GPB and CBOR are so compact, it’s easy to fit into a UDP packet. But for JSON or XML, it is possible that even one leaf will exceed the UDP boundary.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-------------------------------------------------------------+-+
|            Fragment Number                                  |L|
+-------------------------------------------------------------+-+
```

Fig. 5 Fragmentation Option Format

The Fragmentation Option is available in the message header when the fragmentation flag is set to 1. The option contains:

Fragment Number: indicates the sequence number of the current fragment.

L: is a flag to indicate whether the current fragment is the last one. When 0 is set, current fragment is not the last one, hence more fragments are expected. When 1 is set, current fragment is the last one.

4.4. Data Encoding

Subscribed data can be encoded in GPB, CBOR, XML or JSON format. It is conceivable that additional encodings may be supported as options in the future. This can be accomplished by augmenting the subscription data model with additional identity statements used to refer to requested encodings.

Implementation may support different encoding method per subscription. When bundled notifications is supported between the publisher and the receiver, only subscribed notifications with the same encoding can be bundled as one message.

5. Using DTLS to Secure UPC

The Datagram Transport Layer Security (DTLS) protocol [RFC6347] is designed to meet the requirements of applications that need secure datagram transport.

DTLS can be used as a secure transport to counter all the primary threats to UDP based Publication Channel:

- Confidentiality to counter disclosure of the message contents.

- Integrity checking to counter modifications to a message on a hop-by-hop basis.

- Server or mutual authentication to counter masquerade.

In addition, DTLS also provides:

- A cookie exchange mechanism during handshake to counter Denial of Service attacks.

- A sequence number in the header to counter replay attacks.

5.1. Transport

As shown in Figure 6, the DTLS is layered next to the UDP transport is to provide reusable security and authentication functions over UDP. No DTLS extension is required to enable UPC messages over DTLS.

```
+------------------------------+
|         UPC Message          |
+------------------------------+
|           DTLS               |
+------------------------------+
|           UDP                |
+------------------------------+
|           IP                 |
+------------------------------+
```

Fig. 6: Protocol Stack for DTLS secured UPC

The application implementer will map a unique combination of the remote address, remote port number, local address, and local port number to a session.

Each UPC message is delivered by the DTLS record protocol, which assigns a sequence number to each DTLS record. Although the DTLS implementer may adopt a queue mechanism to resolve reordering, it may not assure that all the messages are delivered in order when mapping on the UDP transport.

Since UDP is an unreliable transport, with DTLS, an originator or relay may not realize that a collector has gone down or lost its DTLS connection state, so messages may be lost.

The DTLS record has its own sequence number, the encryption and decryption will done by DTLS layer, UPC Message layer will not concern this.

5.2. Port Assignment

The Publisher is always a DTLS client, and the Receiver is always a DTLS server. The Receivers MUST support accepting UPC Messages on the UDP port PORT-Y, but MAY be configurable to listen on a different port. The Publisher MUST support sending UPC messages to the UDP port PORT-Y, but MAY be configurable to send messages to a different port. The Publisher MAY use any source UDP port for transmitting messages.

5.3. DTLS Session Initiation

The Publisher initiates a DTLS connection by sending a DTLS Client Hello to the Receiver. Implementations MUST support the denial of service countermeasures defined by DTLS. When these countermeasures are used, the Receiver responds with a DTLS Hello Verify Request containing a cookie. The Publisher responds with a DTLS Client Hello containing the received cookie, which initiates the DTLS handshake.

The Publisher MUST NOT send any UPC messages before the DTLS handshake has successfully completed.

Implementations MUST support DTLS 1.0 [RFC4347] and MUST support the mandatory to implement cipher suite, which is TLS_RSA_WITH_AES_128_CBC_SHA [RFC5246] as specified in DTLS 1.0. If additional cipher suites are supported, then implementations MUST NOT negotiate a cipher suite that employs NULL integrity or authentication algorithms.

Where privacy is REQUIRED, then implementations must either negotiate a cipher suite that employs a non-NULL encryption algorithm or else achieve privacy by other means, such as a physically secured network.

5.4. Sending Data

All UPC messages MUST be sent as DTLS "application_data". It is possible that multiple UPC messages be contained in one DTLS record, or that a publication message be transferred in multiple DTLS records. The application data is defined with the following ABNF [RFC5234] expression:

```
APPLICATION-DATA = 1*UPC-FRAME

UPC-FRAME = MSG-LEN SP UPC-MSG

MSG-LEN = NONZERO-DIGIT *DIGIT

SP = %d32

NONZERO-DIGIT = %d49-57

DIGIT = %d48 / NONZERO-DIGIT
```

UPC-MSG is defined in section 5.2.

5.5. Closure

A Publisher MUST close the associated DTLS connection if the connection is not expected to deliver any UPC Messages later. It MUST send a DTLS close_notify alert before closing the connection. A Publisher (DTLS client) MAY choose to not wait for the Receiver's close_notify alert and simply close the DTLS connection. Once the Receiver gets a close_notify from the Publisher, it MUST reply with a close_notify.

When no data is received from a DTLS connection for a long time (where the application decides what "long" means), Receiver MAY close the connection. The Receiver (DTLS server) MUST attempt to initiate an exchange of close_notify alerts with the Publisher before closing the connection. Receivers that are unprepared to receive any more data MAY close the connection after sending the close_notify alert.

Although closure alerts are a component of TLS and so of DTLS, they, like all alerts, are not retransmitted by DTLS and so may be lost over an unreliable network.

6. Congestion Control

Congestion control mechanisms that respond to congestion by reducing traffic rates and establish a degree of fairness between flows that share the same path are vital to the stable operation of the Internet [RFC2914]. While efficient, UDP has no build-in congestion control mechanism. Because streaming telemetry can generate unlimited amounts of data, transferring this data over UDP is generally problematic. It is not recommended to use the UDP based publication channel over congestion-sensitive network paths. The only environments where the UDP based publication channel MAY be used are managed networks. The deployments require the network path has been explicitly provisioned for the UDP based publication channel through traffic engineering mechanisms, such as rate limiting or capacity reservations.

7. A YANG Data Model for Management of UPC

The YANG model defined in Section 9 has two leafs augmented into one place of Sub-Notif [I-D.ietf-netconf-subscribed-notifications], plus one identities.

```
  module: ietf-upc-subscribed-notifications
     augment /sn:subscriptions/sn:subscription/sn:receivers/sn:receiver:
       +--rw address?   inet:ip-address
       +--rw port?      inet:port-number
```

8. YANG Module

```
<CODE BEGINS> file "ietf-upc-subscribed-notifications@2018-10-19.yang"
module ietf-upc-subscribed-notifications {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-upc-subscribed-notifications";
  prefix upcsn;
  import ietf-subscribed-notifications {
    prefix sn;
  }
  import ietf-inet-types {
```

```
    prefix inet;
  }

  organization "IETF NETCONF (Network Configuration) Working Group";
  contact
    "WG Web:   <http:/tools.ietf.org/wg/netconf/>
     WG List:  <mailto:netconf@ietf.org>

     Editor:   Guangying Zheng
               <mailto:zhengguangying@huawei.com>

     Editor:   Tianran Zhou
               <mailto:zhoutianran@huawei.com>

     Editor:   Alexander Clemm
               <mailto:alexander.clemm@huawei.com>";


  description
    "Defines UDP Publish Channel as a supported transport for subscribed
    event notifications.

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

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

    This version of this YANG module is part of RFC XXXX; see the RFC

    itself for full legal notices.";

  revision 2018-10-19 {
    description
      "Initial version";
    reference
      "RFC XXXX: UDP based Publication Channel for Streaming Telemetry";
  }

  identity upc {
    base sn:transport;
    description
      "UPC is used as transport for notification messages and state
       change notifications.";
  }
```

```
  grouping target-receiver {
    description
      "Provides a reusable description of a UPC target receiver.";
    leaf address {
      type inet:ip-address;
      description
        "Ip address of target upc receiver, which can be IPv4 address or
         IPV6 address.";
    }
    leaf port {
      type inet:port-number;
      description
        "Port number of target UPC receiver, if not specify, system
         should use default port number.";
    }
  }

  augment "/sn:subscriptions/sn:subscription/sn:receivers/sn:receiver" {
    description
      "This augmentation allows UPC specific parameters to be
       exposed for a subscription.";
    uses target-receiver;
  }
}
<CODE ENDS>
```

9. IANA Considerations

This RFC requests that IANA assigns three UDP port numbers in the "Registered Port Numbers" range with the service names "upc" and "upc-dtls". These ports will be the default ports for the UDP based Publication Channel for NETCONF and RESTCONF. Below is the registration template following the rules in [RFC6335].

Service Name: upc

Transport Protocol(s): UDP

Assignee: IESG <iesg@ietf.org>

Contact: IETF Chair <chair@ietf.org>

Description: UDP based Publication Channel

Reference: RFC XXXX

Port Number: PORT-X

Service Name: upc-dtls

Transport Protocol(s): UDP

Assignee: IESG <iesg@ietf.org>

Contact: IETF Chair <chair@ietf.org>

Description: UDP based Publication Channel (DTLS)

Reference: RFC XXXX

Port Number: PORT-Y

IANA is requested to assign a new URI from the IETF XML Registry [RFC3688]. The following URI is suggested:

URI: urn:ietf:params:xml:ns:yang:ietf-upc-subscribed-notifications
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.

This document also requests a new YANG module name in the YANG Module Names registry [RFC7950] with the following suggestion:

```
name: ietf-upc-subscribed-notifications
namespace: urn:ietf:params:xml:ns:yang:ietf-upc-subscribed-notifications
prefix: upcsn
reference: RFC XXXX
```

10. Security Considerations

TBD

11. Acknowledgements

The authors of this documents would like to thank Eric Voit, Tim Jenkins, and Huiyang Yang for the initial comments.

12. References

12.1. Normative References

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

[RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, DOI 10.17487/RFC2914, September 2000, <https://www.rfc-editor.org/info/rfc2914>.

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

[RFC4347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security", RFC 4347, DOI 10.17487/RFC4347, April 2006, <https://www.rfc-editor.org/info/rfc4347>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

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

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <https://www.rfc-editor.org/info/rfc6335>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <https://www.rfc-editor.org/info/rfc6347>.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <https://www.rfc-editor.org/info/rfc7049>.

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

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

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

12.2. Informative References

[I-D.ietf-netconf-netconf-event-notifications]
Voit, E., Clemm, A., Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Dynamic subscription to YANG Events and Datastores over NETCONF", draft-ietf-netconf-netconf-event-notifications-17 (work in progress), February 2019.

[I-D.ietf-netconf-notification-messages]
Voit, E., Birkholz, H., Bierman, A., Clemm, A., and T. Jenkins, "Notification Message Headers and Bundles", draft-ietf-netconf-notification-messages-05 (work in progress), February 2019.

[I-D.ietf-netconf-restconf-notif]
Voit, E., Rahman, R., Nilsen-Nygaard, E., Clemm, A., and A. Bierman, "Dynamic subscription to YANG Events and Datastores over RESTCONF", draft-ietf-netconf-restconf-notif-13 (work in progress), February 2019.

[I-D.ietf-netconf-subscribed-notifications]
Voit, E., Clemm, A., Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Subscription to YANG Event Notifications", draft-ietf-netconf-subscribed-notifications-23 (work in progress), February 2019.

[I-D.zhou-netconf-multi-stream-originators]
Zhou, T., Zheng, G., Voit, E., Clemm, A., and A. Bierman, "Subscription to Multiple Stream Originators", draft-zhou-netconf-multi-stream-originators-03 (work in progress), October 2018.

12.3. URIs

[1] https://developers.google.com/protocol-buffers/

Appendix A. Change Log

(To be removed by RFC editor prior to publication)

A.1. draft-ietf-zheng-udp-pub-channel-00 to v00

- o Modified the message header format.
- o Added a section on the Authentication Option.
- o Cleaned up the text and removed unnecessary TBDs.

A.2. v01

- o Removed the detailed description on distributed data collection mechanism from this document. Mainly focused on the description of a UDP based publication channel for telemetry use.
- o Modified the message header format.

A.2. v02

- o Add the section on the transport mechanism.
- o Modified the fixed message header format.
- o Add the fragmentation option for the message header.

A.2. v03

- o Clarify term through the document.
- o Add a section on DTLS support.

A.2. v04

- o Add a section on UPC subscription model.

A.2. v05

- o Remove the redundant solution overview section and refer to the multi stream originator draft.

Authors' Addresses

Guangying Zheng
Huawei
101 Yu-Hua-Tai Software Road
Nanjing, Jiangsu
China

Email: zhengguangying@huawei.com

Tianran Zhou
Huawei
156 Beiqing Rd., Haidian District
Beijing
China

Email: zhoutianran@huawei.com

Alexander Clemm
Huawei
2330 Central Expressway
Santa Clara, California
USA

Email: alexander.clemm@huawei.com

NETCONF A. Clemm
Internet-Draft Futurewei
Intended status: Standards Track E. Voit
Expires: November 22, 2019 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

Copyright Notice

Table of Contents

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:

- Polling incurs significant latency. This latency prohibits many types of application.
- 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.
- Polling requests may undergo slight fluctuations, resulting in intervals of different lengths. The resulting data is difficult to calibrate and compare.
- For applications that monitor for changes, many remote polling cycles place unwanted and ultimately wasteful load on the network, devices, and applications, particularly when changes occur only infrequently.

A more effective alternative to polling is for an application to receive automatic and continuous updates from a targeted subset of a datastore. Accordingly, there is a need for a service that allows applications to subscribe to updates from a datastore and that enables the server (also referred to as publisher) to push and in effect stream those updates. The requirements for such a service have been documented in [RFC7923].

This document defines a corresponding solution that is built on top of "Custom Subscription to Event Streams" [I-D.draft-ietf-netconf-subscribed-notifications]. Supplementing that work are YANG data model augmentations, extended RPCs, and new datastore specific update notifications. Transport options for [I-D.draft-ietf-netconf-subscribed-notifications] will work seamlessly with this solution.

2. Definitions and Acronyms

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:

- Datastore node: A node in the instantiated YANG data tree associated with a datastore. In this document, datastore nodes are often also simply referred to as "objects"

- Datastore node update: A data item containing the current value of a datastore node at the time the datastore node update was created, as well as the path to the datastore node.

- Datastore subscription: A subscription to a stream of datastore node updates.

- Datastore subtree: A datastore node and all its descendant datastore nodes

- On-change subscription: A datastore subscription with updates that are triggered when changes in subscribed datastore nodes are detected.

- Periodic subscription: A datastore subscription with updates that are triggered periodically according to some time interval.

- Selection filter: Evaluation and/or selection criteria, which may be applied against a targeted set of objects.

- Update record: A representation of one or more datastore node updates. In addition, an update record may contain which type of update led to the datastore node update (e.g., whether the datastore node was added, changed, deleted). Also included in the update record may be other metadata, such as a subscription id of the subscription as part of which the update record was generated. In this document, update records are often also simply referred to as "updates".

- Update trigger: A mechanism that determines when an update record needs to be generated.

- 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:

- Specification of selection filters which identify targeted YANG datastore nodes and/or datastore subtrees for which updates are to be pushed.

- 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-datasore-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.

```
                       +-----------------+      +--------------------+
  push-update or -->   | datastore node  | yes  | add datastore node |
 push-change-update    | access allowed? | ---> | to update record   |
                       +-----------------+      +--------------------+
```

Figure 5: Updated [RFC8341] access control for push updates

A publisher MUST allow for the possibility that a subscription's selection filter references non-existent data or data that a receiver is not allowed to access. Such support permits a receiver the ability to monitor the entire lifecyle of some datastore tree without needing to explicitly enumerate every individual datastore node. If, after access control has been applied, there are no objects remaining in an update record, then (in case of a periodic subscription) only a single empty "push-update" notification MUST be sent. Empty "push-change-update" messages (in case of an on-change subscription) MUST NOT be sent. This is required to ensure that clients cannot surreptitiously monitor objects that they do not have access to via carefully crafted selection filters. By the same token, changes to objects that are filtered MUST NOT affect any dampening intervals.

A publisher MAY choose to reject an establish-subscription request which selects non-existent data or data that a receiver is not allowed to access. As reason, the error identity "unchanging-selection" SHOULD be returned. In addition, a publisher MAY choose to terminate a dynamic subscription or suspend a configured receiver when the authorization privileges of a receiver change, or the access controls for subscribed objects change. In that case, the publisher SHOULD include the error identity "unchanging-selection" as reason when sending the "subscription-terminated" respectively "subscription-suspended" notification. Such a capability enables the publisher to avoid having to support continuous and total filtering of a subscription's content for every update record. It also reduces the possibility of leakage of access-controlled objects.

If read access into previously accessible nodes has been lost due to a receiver permissions change, this SHOULD be reported as a patch "delete" operation for on-change subscriptions. If not capable of handling such receiver permission changes with such a "delete", publisher implementations MUST force dynamic subscription re-establishment or configured subscription re-initialization so that appropriate filtering is installed.

3.10. On-Change Notifiable Datastore Nodes

In some cases, a publisher supporting on-change notifications may not be able to push on-change updates for some object types. Reasons for this might be that the value of the datastore node changes frequently (e.g., [RFC8343]'s in-octets counter), that small object changes are frequent and meaningless (e.g., a temperature gauge changing 0.1 degrees), or that the implementation is not capable of on-change notification for a particular object.

In those cases, it will be important for client applications to have a way to identify for which objects on-change notifications are supported and for which ones they are not supported. Otherwise client applications will have no way of knowing whether they can indeed rely on their on-change subscription to provide them with the change updates that they are interested in. In other words, if implementations do not provide a solution and do not support comprehensive on-change notifiability, clients of those implementations will have no way of knowing what their on-change subscription actually covers.

Implementations are therefore strongly advised to provide a solution to this problem. 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 (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 ageneral-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 "modifiy-subscription" RPC response. This MUST be sent if hints are included.

3. YANG-data sn:delete-subscription-error
   This MUST be returned if information identifying the reason for an RPC error has not been placed elsewhere within the transport portion of a failed "delete-subscription" or "kill-subscription" RPC response.

4. YANG-data resync-subscription-error
   This MUST be returned if information identifying the reason for an RPC error has not been placed elsewhere within the transport portion of a failed "resync-subscription" RPC response.

A.2. Notifications of Failure

A subscription may be unexpectedly terminated or suspended independent of any RPC or configuration operation. In such cases, indications of such a failure MUST be provided. To accomplish this, 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    period-unsupported
 unchanging-selection          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.

- Error model moved to match existing implementations
- On-change notifiable removed, how to do this is implementation specific.
- NMDA model supported. Non NMDA version at https://github.com/netconf-wg/yang-push/

v10 - v11

- Promise model reference added.
- Error added for no-such-datastore
- Inherited changes from subscribed notifications (such as optional feature definitions).
- scrubbed the examples for proper encodings

v09 - v10

- Returned to the explicit filter subtyping of v00-v05
- identityref to ds:datastore made explicit
- Returned ability to modify a selection filter via RPC.

v08 - v09

- Minor tweaks cleaning up text, removing appendicies, and making reference to revised-datastores.
- Subscription-id (now:id) optional in push updates, except when encoded in RFC5277, Section 4 one-way notification.
- Finished adding the text descibing the resync subscription RPC.
- Removed relationships to other drafts and future technology appendicies as this work is being explored elsewhere.
- Deferred the multi-line card issue to new drafts
- Simplified the NACM interactions.

v07 - v08

- 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 K. Watsen
Internet-Draft Juniper Networks
Intended status: Standards Track M. Abrahamsson
Expires: July 19, 2019 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

Copyright Notice

Table of Contents

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 devices 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 posses 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 maximumly 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.
_sztp._tcp.example.com. 1800 IN SRV 1 0 443 sztp2.example.com.
_sztp._tcp.example.com. 1800 IN SRV 2 0 443 sztp3.example.com.
_sztp._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

```
+-------------------------------------------------------------+
|                           <device>                          |
|                                                             |
|  +-------------------------------------------------------+  |
|  |                  <read/write storage>                 |  |
|  |                                                       |  |
|  | 1. flag to enable SZTP bootstrapping set to "true"    |  |
|  +-------------------------------------------------------+  |
|                                                             |
|  +-------------------------------------------------------+  |
|  |                  <read-only storage>                  |  |
|  |                                                       |  |
|  | 2. TLS client cert & related intermediate certificates |  |
|  | 3. list of trusted well-known bootstrap servers       |  |
|  | 4. list of trust anchor certs for bootstrap servers   |  |
|  | 5. list of trust anchor certs for ownership vouchers  |  |
|  +-------------------------------------------------------+  |
|                                                             |
|    +---------------------------------------------------+    |
|    |                <secure storage>                   |    |
|    |                                                   |    |
|    |  6. private key for TLS client certificate        |    |
|    |  7. private key for decrypting SZTP artifacts     |    |
|    +---------------------------------------------------+    |
|                                                             |
+-------------------------------------------------------------+
```

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 posses 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.

```
      Power On
          |
          v                            No
   1. SZTP bootstrapping configured ------> Boot normally
          |
          | Yes
          v
   2. For each supported source of bootstrapping data,
      try to load bootstrapping data from the source
          |
          |
          v                                Yes
   3. Able to bootstrap from any source? -----> Run with new config
          |
          | No
          v
   4. Loop back to Step 1.
```

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 processes 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 a 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-bootstrappi\
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-bootstrappi\
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
         bootstap 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.

- 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>.

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.

```
                                                    +----------+
                                                    |Deployment|
                                                    | Specific |
   +------+                                         |Bootstrap |
   |Device|                                         |  Server  |
   +------+                                         +----------+
      |                                                   |
      | 1. "HTTPS" Request ("signed-data-preferred", nonce)|
      |-------------------------------------------------->|
      | 2. "HTTPS" Response (signed redirect information)  |
      |<--------------------------------------------------|
      |                                                   |
      |                                                   |
      | 3. HTTPS Request (os-name=xyz, os-version=123, etc.)|
      |-------------------------------------------------->|
      | 4. HTTPS Response (unsigned onboarding information |
      |<--------------------------------------------------|
      |                                                   |
```

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.

```
                                  +-----------+
   +------------+                 |Prospective|                    +---+
   |Manufacturer|                 |   Owner   |                    |NMS|
   +------------+                 +-----------+                    +---+
         |                              |                            |
         |                              |                            |
         |  1. initiate enrollment      |                            |
         #<-----------------------------|                            |
         #                              |                            |
         #                              |                            |
         #     IDevID trust anchor      |                            |
         #----------------------------->#  set IDevID trust anchor   |
         #                              #--------------------------->|
         #                              |                            |
         #     bootstrap server         |                            |
         #     account credentials      |                            |
         #----------------------------->#  set credentials           |
         |                              #--------------------------->|
         |                              |                            |
         |                              |                            |
         |  2. set owner certificate trust anchor                    |
         |<----------------------------------------------------------|
         |                              |                            |
         |                              |                            |
         |  3. place device order       |                            |
         |<-----------------------------#  model devices             |
         |                              #--------------------------->|
         |                              |                            |
         |  4. ship devices and send    |                            |
         |     device identifiers and   |                            |
         |     ownership vouchers       |                            |
         |----------------------------->#  set device identifiers    |
         |                              #  and ownership vouchers    |
         |                              #--------------------------->|
         |                              |                            |
```

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 is 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.

```
              +----------+ +------------+
              |Deployment| |Manufacturer| +------+ +------+
              | Specific | |   Hosted   | | Local| | Local| +---------+
        +---+ |Bootstrap | | Bootstrap  | |  DNS | | DHCP | |Removable|
        |NMS| |  Server  | |   Server   | |Server| |Server| | Storage |
        +---+ +----------+ +------------+ +------+ +------+ +---------+
          |         |             |            |        |          |
 1.       |         |             |            |        |          |
 activate |         |             |            |        |          |
 modeled  |         |             |            |        |          |
 device   |         |             |            |        |          |
 -------> |         |             |            |        |          |
          | 2. (optional)         |            |        |          |
          |    configure          |            |        |          |
          |    bootstrap          |            |        |          |
          |    server             |            |        |          |
          |-------> |             |            |        |          |
          |         |             |            |        |          |
          | 3. (optional) configure            |        |          |
          |    bootstrap server   |            |        |          |
          |---------------------> |            |        |          |
          |         |             |            |        |          |
          |         |             |            |        |          |
          | 4. (optional) configure DNS server |        |          |
          |----------------------------------->|        |          |
          |         |             |            |        |          |
          |         |             |            |        |          |
          | 5. (optional) configure DHCP server         |          |
          |-------------------------------------------->|          |
          |         |             |            |        |          |
          |         |             |            |        |          |
          | 6. (optional) store bootstrapping artifacts on media   |
          |------------------------------------------------------->|
          |         |             |            |        |          |
          |         |             |            |        |          |
```

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.

```
                                                     +----------+
                                      +-----------+  |Deployment|
                                      | Source of |  | Specific |
   +------+                           | Bootstrap |  |Bootstrap |  +---+
   |Device|                           |   Data    |  |  Server  |  |NMS|
   +------+                           +-----------+  +----------+  +---+
      |                                     |              |         |
      |                                     |              |         |
      | 1. if SZTP bootstrap service        |              |         |
      |    is not enabled, then exit.       |              |         |
      |                                     |              |         |
      | 2. for each source supported, check |              |         |
      |    for bootstrapping data.          |              |         |
      |------------------------------------>|              |         |
      |                                     |              |         |
      | 3. if onboarding information found, |              |         |
      |    initialize self and, only if     |              |         |
      |    source is a trusted bootstrap    |              |         |
      |    server, send progress reports.   |              |         |
      |------------------------------------>#              |         |
      |                                     # webhook      |         |
      |                                     #----------------------->|
      |                                                    |         |
      | 4. else if redirect-information found, for each    |         |
      |    bootstrap server specified, check for data.     |         |
      |-+------------------------------------------------->|         |
      | |                                                  |         |
      | | if more redirect-information is found, recurse   |         |
      | | (not depicted), else if onboarding information   |         |
      | | found, initialize self and post progress reports |         |
      | +------------------------------------------------->#         |
      |                                                    # webhook |
      |                                                    #-------->|
      |
      | 5. retry sources and/or wait for manual provisioning.
      |
```

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

- Major structural update; the essence is the same. Most every section was rewritten to some degree.
- Added a Use Cases section
- Added diagrams for "Actors and Roles" and "NMS Precondition" sections, and greatly improved the "Device Boot Sequence" diagram
- Removed support for physical presence or any ability for configlets to not be signed.
- Defined the Conveyed Information DHCP option
- Added an ability for devices to also download images from configuration servers
- Added an ability for configlets to be encrypted
- Now configuration servers only have to support HTTP/S - no other schemes possible

D.2. 00 to 01

- Added boot-image and validate-owner annotations to the "Actors and Roles" diagram.
- Fixed 2nd paragraph in section 7.1 to reflect current use of anyxml.
- Added encrypted and signed-encrypted examples
- Replaced YANG module with XSD schema
- Added IANA request for the Conveyed Information DHCP Option
- Added IANA request for media types for boot-image and configuration

D.3. 01 to 02

- 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).

- combined the information and signature PKCS #7 structures into a single PKCS #7 structure.

- moved the certificate-revocations into the owner-certificate's PKCS #7 structure.

- eliminated support for voucher-revocations, to reflect the voucher-draft's switch from revocations to renewals.

D.15. 13 to 14

- Renamed "bootstrap information" to "onboarding information".

- Rewrote DHCP sections to address the packet-size limitation issue, as discussed in Chicago.

- Added Ian as an author for his text-contributions to the DHCP sections.

- Removed the Guiding Principles section.

D.16. 14 to 15

- Renamed action "notification" to "update-progress" and, likewise "notification-type" to "update-type".

- Updated examples to use "base64encodedvalue==" for binary values.

- Greatly simplified the "Artifact Groupings" section, and moved it as a subsection to the "Artifacts" section.

- Moved the "Workflow Overview" section to the Appendix.

- Renamed "bootstrap information" to "update information".

- Removed "Other Considerations" section.

- Tons of editorial updates.

D.17. 15 to 16

- tweaked language to refer to "initial state" rather than "factory default configuration", so as accommodate white-box scenarios.

- 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.

- Added an "signed-data-preferred" parameter to "get-bootstrapping-data" RPC.

- Added the "ietf-zerotouch-device" module.

- Lots of small updates.

D.20. 18 to 19

- 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

- Now references I-D.ietf-netmod-yang-tree-diagrams.

- Fixed tree-diagrams in Section 2 to always reflect current YANG (now they are now dynamically generated).

- The "redirect-information" container's "trust-anchor" is now a CMS structure that can contain a chain of certificates, rather than a single certificate.

- The "onboarding-information" container's support for image verification reworked to be extensible.

- Added a reference to the "Device Details" section to the new example-device-data-model module.

- Clarified that the device must always pass its IDevID certificate, even for untrusted bootstrap servers.

- 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.

- For the get-bootstrapping-data RPC's input, removed the "remote-id" and "circuit-id" fields, and added a "hw-model" field.

- Improved DHCP error handling text.

- 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 onboboarding 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 conifguration 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

- Added a couple features to the bootstrap-server module.

D.29. 27 to 28

- Modified DNS section to no longer reference DNS-SD (now just plain TXT and SRV lookups, via multicast or unicast.

- Registers "_sztp" in the DNS Underscore Global Scoped Entry Registry.

- 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

NETCONF T. Zhou
Internet-Draft G. Zheng
Intended status: Standards Track Huawei
Expires: January 8, 2020 E. Voit
Cisco Systems
A. Clemm
Futurewai
A. Bierman
YumaWorks
July 07, 2019

Subscription to Multiple Stream Originators
draft-zhou-netconf-multi-stream-originators-06

Abstract

This document describes the distributed data export mechanism that allows multiple data streams to be managed using a single subscription. Specifically, multiple data streams are pushed directly to the collector without passing through a broker for internal consolidation.

Requirements Language

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

Status of This Memo

Table of Contents

1. Introduction

Streaming telemetry refers to sending a continuous stream of operational data from a device to a remote receiver. This provides an ability to monitor a network from remote and to provide network analytics. Devices generate telemetry data and push that data to a collector for further analysis. By streaming the data, much better performance, finer-grained sampling, monitoring accuracy, and bandwidth utilization can be achieved than with polling-based alternatives.

YANG-Push [I-D.ietf-netconf-yang-push] defines a transport-independent subscription mechanism for datastore updates, in which a subscriber can subscribe to a stream of datastore updates from a server, or update provider. The current design involves subscription to a single push server. This conceptually centralized model encounters efficiency limitations in cases where the data sources are themselves distributed, such as line cards in a piece of network equipment. In such cases, it will be a lot more efficient to have each data source (e.g., each line card) originate its own stream of updates, rather than requiring updates to be tunneled through a central server where they are combined. What is needed is a distributed mechanism that allows to directly push multiple individual data substreams, without needing to first pass them through an additional processing stage for internal consolidation, but still allowing those substreams to be managed and controlled via a single subscription.

This document will describe such distributed data collection mechanism and how it can work by extending existing YANG-Push mechanism. The proposal is general enough to fit many scenarios.

2. Use Cases

2.1. Use Case 1: Data Collection from Devices with Main-board and Line-cards

For data collection from devices with main-board and line-cards, existing YANG-Push solutions consider only one push server typically reside in the main board. As shown in the following figure, data are collected from line cards and aggregate to the main board as one consolidated stream. So the main board can easily become the performance bottle-neck. The optimization is to apply the distributed data collection mechanism which can directly push data from line cards to a collector. On one hand, this will reduce the cost of scarce compute and memory resources on the main board for data processing and assembling. On the other hand, distributed data push can off-load the streaming traffic to multiple interfaces.

```
                  +-------------------------------------+
                  |              collector              |
                  +------^-----------^-----------^------+
                         |           |           |
                         |           |           |
                  +-------------------------------------+
                  |      |           |           |      |
                  |      |     +-----+------+    |      |
                  |      |     | main board |    |      |
                  |      |     +--^-----^---+    |      |
                  |      |        |     |        |      |
                  |      |    +---+     +---+    |      |
                  |      |    |             |    |      |
                  | +----+----+---+     +---+----+----+ |
                  | | line card 1 |     | line card 2 | |
                  | +-------------+     +-------------+ |
                  |               device                |
                  +-------------------------------------+
```

Fig. 1 Data Collection from Devices with Main-board and Line-cards

2.2. Use Case 2: IoT Data Collection

In the IoT data collection scenario, as shown in the following figure, collector usually cannot access to IoT nodes directly, but is isolated by the border router. So the collector subscribes data from the border router, and let the border router to disassemble the subscription to corresponding IoT nodes. The border router is typically the traffic convergence point. It’s intuitive to treat the border router as a broker assembling the data collected from the IoT nodes and forwarding to the collector[I-D.ietf-core-coap-pubsub]. However, the border router is not so powerful on data assembling as a network device. It’s more efficient for the collector, which may be a server or even a cluster, to assemble the subscribed data if possible. In this case, push servers that reside in IoT nodes can stream data to the collector directly while traffic only passes through the border router.

```
+--------------------------------+
|            collector           |
+---^-----------^------------^--+
    |           |            |
    |           |            |
    |           |            |
    |   +-------+--------+   |
    |   |  border router |   |
    |   +----^------^----+   |
    |        |      |        |
    |        |      |        |
    |    +---+      +---+    |
    |    |              |    |
+---+----+---+      +---+----+---+
| IoT node 1 |      | IoT node 2 |
+------------+      +------------+
```

Fig. 2 IoT Data Collection

3. Terminologies

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

Receiver: is the target for the data publication.

Publisher: pushes data to the receiver according to the subscription information.

Subscription Server: which manages capabilities that it can provide to the subscriber.

Global Subscription: the subscription requested by the subscriber. It may be decomposed into multiple Component Subscriptions.

Component Subscription: is the subscription that defines the data from each individual telemetry source which is managed and controlled by a single Subscription Server.

Global Capability: is the overall subscription capability that the group of Publishers can expose to the Subscriber.

Component Capability: is the subscription capability that each Publisher can expose to the Subscriber.

Master Publication Channel: the session between the Master Publisher and the Receiver.

Agent Publication Channel: the session between the Agent Publisher and the Receiver.

4. Solution Overview

All the use cases described in the previous section are very similar on the data subscription and publication mode, hence can be abstracted to the following generic distributed data collection framework, as shown in the following figure.

A Collector usually includes two components,

- o the Subscriber generates the subscription instructions to express what and how the collector want to receive the data;
- o the Receiver is the target for the data publication.

For one subscription, there may be one to many receivers. And the subscriber does not necessarily share the same address with the receivers.

In this framework, the Publisher pushes data to the receiver according to the subscription information. The Publisher has the Master role and the Agent role. Both the Master and the Agent include the Subscription Server which actually manages capabilities that it can provide to the subscriber.

The Master knows all the capabilities that the attached Agents and itself can provide, and exposes the Global Capability to the Collector. The Collector cannot see the Agents directly, so it will only send the Global Subscription information to the Master. The Master disassembles the Global Subscription to multiple Component Subscriptions, each involving data from a separate telemetry source. The Component Subscriptions are then distributed to the corresponding Agents.

When data streaming, the Publisher collects and encapsulates the packets per the Component Subscription, and pushes the piece of data which can serve directly to the designated data Collector. The Collector is able to assemble many pieces of data associated with one Global Subscription, and can also deduce the missing pieces of data.

```
                    +--------------------------------------+
                    |      Collector      |--------------+ |
                    |                    +-------------+ | |
                    |   +------------+   || Receiver   | | |
                    |   | Subscriber |   |---------------+ |
                    |   +-^----+-----+   +---^---------^   |
                    |     |    |             |         |   |
                    +--------------------------------------+
               Global     |    |Global       |Push     |
               Capability |    |Subscription |         |
                    +------------------------+------+  |
                    |     |    | Publisher(Master)  |  |
                    |  +--+----v------+             |  |
                    |  | Subscription |             |  |
                    |  | Server       |             |  |
                    |  +--^----+------+             |  |
                    |     |    |                    |  |
                    +-------------------------------+  |
               Component  |    | Component             |Push
               Capability |    | Subscription          |
                    +-------------------------------+  |
                    |     |    | Publisher(Agent)   |  |
                    |  +--+----v------+             |  |
                    |  | Component    |             |  |
                    |  | Subscription |             +--+
                    |  | Server       |             |
                    |  +--------------+             |
                    +-------------------------------+
```

Fig. 3 The Generic Distributed Data Collection Framework

Master and Agents may interact with each other in several ways:

- Agents need to have a registration or announcement handshake with the Master, so the Master is aware of them and of life-cycle events (such as Agent appearing and disappearing).

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

- The Master relays the component subscriptions to the Agents.

- The Agents indicate status of Component Subscriptions to the Master. The status of the overall subscription is maintained by the Master. The Master is also responsible for notifying the subscriber in case of any problems of Component Subscriptions.

Any technical mechanisms or protocols used for the coordination of operational information between Master and Agent is out-of-scope of the solution. We will need to instrument the results of this coordination on the Master Node.

5. Subscription Decomposition

Since Agents are invisible to the Collector, the Collector can only subscribe to the Master. This requires the Master to:

1. expose the Global Capability that can be served by multiple Publishers;

2. disassemble the Global Subscription to multiple Component Subscriptions, and distribute them to the corresponding telemetry sources;

3. notify on changes when portions of a subscription moving between different Agents over time.

To achieve the above requirements, the Master need a Global Capability description which is typically the YANG [RFC7950] data model. This global YANG model is provided as the contract between the Master and the Collector. Each Agent associating with the Master owns a local YANG model to describe the Component Capabilities which it can serve as part of the Global Capability. All the Agents need to know the namespace associated with the Master.

The Master also need a data structure, typically a Resource-Location Table, to keep track of the mapping between the resource and the corresponding location of the Subscription Server which commits to serve the data. When a Global Subscription request arrives, the Master will firstly extract the filter information from the request. Consequently, according to the Resource-Location Table, the Global Subscription can be disassembled into multiple Component Subscriptions, and the corresponding location can be associated.

The decision whether to decompose a Global Subscription into multiple Component Subscriptions rests with the Resource-Location Table. A Master can decide to not decompose a Global Subscription at all and push a single stream to the receiver, because the location information indicates the Global Subscription can be served locally by the Master. Similarly, it can decide to entirely decompose a Global Subscription into multiple Component Subscriptions that each push their own streams, but not from the Master. It can also decide to decompose the Global Subscription into several Component Subscriptions and retain some aspects of the Global Subscription itself, also pushing its own stream.

Component Subscriptions belonging to the same Global Subscription MUST NOT overlap. The combination of all Component Subscriptions MUST cover the same range of nodes as the Global Subscription. Also, the same subscription settings apply to each Component Subscription, i.e., the same receivers, the same time periods, the same encodings are applied to each Component Subscription per the settings of the Global Subscription.

Each Component Subscription in effect constitutes a full-fledged subscription, with the following constraints:

o Component subscriptions are system-controlled, i.e. managed by the Master, not by the subscriber.

o Component subscription settings such as time periods, dampening periods, encodings, receivers adopt the settings of their Global Subscription.

o The life-cycle of the Component Subscription is tied to the life-cycle of the Global Subscription. Specifically, terminating/removing the Global Subscription results in termination/removal of Component Subscriptions.

o The Component Subscriptions share the same Subscription ID as the Global Subscription.

6. Publication Composition

The Publisher collects data and encapsulates the packets per the Component Subscription. There are several potential encodings, including XML, JSON, CBOR and GPB. The format and structure of the data records are defined by the YANG schema, so that the composition at the Receiver can benefit from the structured and hierarchical data instance.

The Receiver is able to assemble many pieces of data associated with one subscription, and can also deduce the missing pieces of data. The Receiver recognizes data records associated with one subscription according the Subscription ID. Data records generated per one subscription are assigned with the same Subscription ID.

For the time series data stream, records are produced periodically from each stream originator. The message arrival time varies because of the distributed nature of the publication. The Receiver assembles data generated at the same time period based on the recording time consisted in each data record. In this case, time synchronization is required for all the Publishers.

To check the integrity of the data generated from different Publishers at the same time period, the Message Generator ID [I-D.ietf-netconf-notification-messages]is helpful. This requires the Subscriber to know the number of Component Subscriptions which the Global Subscription is decomposed to. For the dynamic subscription, the output of the "establish-subscription" and "modify-subscription" RPC defined in [I-D.ietf-netconf-subscribed-notifications] MUST include a list of Message Generator IDs to indicate how the Global Subscription is decomposed into several Component Subscriptions. The "subscription-started" and "subscription-modified" notification defined in [I-D.ietf-netconf-subscribed-notifications] MUST also include a list of Message Generator IDs to notify the current Publishers for the corresponding Global Subscription.

7. Subscription State Change Notifications

In addition to sending event records to receivers, the Master MUST also send subscription state change notifications[I-D.ietf-netconf-subscribed-notifications] when events related to subscription management have occurred. All the subscription state change notifications MUST be delivered by the Master Publication Channel which is the session between the Master Publisher and the Receiver.

When the subscription decomposition result changed, the "subscription-modified" notification MUST be sent to indicate the new list of Publishers.

8. YANG Tree

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

9. YANG Module

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

  import ietf-inet-types {
    prefix inet;
  }

  organization "IETF NETCONF (Network Configuration) Working Group";
  contact
    "WG Web:   <http:/tools.ietf.org/wg/netconf/>
     WG List:  <mailto:netconf@ietf.org>

     Editor:   Tianran Zhou
               <mailto:zhoutianran@huawei.com>

     Editor:   Guangying Zheng
               <mailto:zhengguangying@huawei.com>";
```

```
   description
     "Defines augmentation for ietf-subscribed-notifications to enable
      the distributed publication with single subscription.

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

     Redistribution and use in source and binary forms, with or without
     modification, is permitted pursuant to, and subject to the license
     terms contained in, the Simplified BSD License set forth in Section
     4.c of the IETF Trust's Legal Provisions Relating to IETF Documents
     (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 2019-07-07 {
     description
       "Initial version";
     reference
       "RFC XXXX: Subscription to Multiple Stream Originators";
   }

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

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

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

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

       case restconf-access {
```

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

   augment "/sn:subscriptions/sn:subscription" {
     description
       "This augmentation allows the message generators to be exposed
        for a subscription.";

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

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

     uses message-generator-ids;
   }

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

     uses message-generator-ids;
   }

   augment "/sn:establish-subscription/sn:output" {
     description
       "This augmentation allows MSO specific parameters to be
        exposed for a subscription.";

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

```
   augment "/sn:modify-subscription/sn:output" {
     description
       "This augmentation allows MSO specific parameters to be
        exposed for a subscription.";

     uses resource-access-list;
     uses message-generator-ids;
   }
 }
 <CODE ENDS>
```

10. IANA Considerations

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

URI: urn:ietf:params:xml:ns:yang:ietf-multiple-stream-originators

Registrant Contact: The IESG.

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

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

Name: ietf-multiple-stream-originators

Namespace: urn:ietf:params:xml:ns:yang:ietf-multiple-stream-originators

Prefix: mso

Reference: RFC XXXX

11. Transport Considerations

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

12. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC5246].

The NETCONF Access Control Model (NACM) [RFC6536] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

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

o /subscriptions/subscription/message-generator-ids

o /subscriptions/subscription/resource-access-list

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

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

13. Acknowledgements

TBD

14. References

14.1. Normative References

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

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

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

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

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

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <https://www.rfc-editor.org/info/rfc6536>.

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

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

14.2. Informative References

[I-D.ietf-core-coap-pubsub]
Koster, M., Keranen, A., and J. Jimenez, "Publish-Subscribe Broker for the Constrained Application Protocol (CoAP)", draft-ietf-core-coap-pubsub-08 (work in progress), March 2019.

[I-D.ietf-netconf-notification-messages]
Voit, E., Birkholz, H., Bierman, A., Clemm, A., and T. Jenkins, "Notification Message Headers and Bundles", draft-ietf-netconf-notification-messages-05 (work in progress), February 2019.

[I-D.ietf-netconf-subscribed-notifications]
Voit, E., Clemm, A., Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Subscription to YANG Event Notifications", draft-ietf-netconf-subscribed-notifications-26 (work in progress), May 2019.

[I-D.ietf-netconf-yang-push]
Clemm, A. and E. Voit, "Subscription to YANG Datastores", draft-ietf-netconf-yang-push-25 (work in progress), May 2019.

[I-D.mahesh-netconf-https-notif]
Jethanandani, M. and K. Watsen, "An HTTPS-based Transport for Configured Subscriptions", draft-mahesh-netconf-https-notif-00 (work in progress), June 2019.

Appendix A. Examples

A.1. RESTCONF Establishing Dynamic Subscription

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

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

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

Fig. 4 establish-subscription request

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

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

Fig. 5 establish-subscription success

Upon receipt of the successful response, the subscriber GET the provided URIs to start the flow of notification messages.

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

Fig. 6 establish-subscription subsequent POST

A.2. HTTPS Configured Subscription

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

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


```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <receivers
      xmlns="urn:ietf:params:xml:ns:yang:ietf-https-notif">
      <receiver>
        <name>foo</name>
        <channel>
          <tcp-params>
            <remote-address>192.0.2.1</remote-address>
            <remote-port>443</remote-port>
            <local-address>192.0.3.1</local-address>
            <local-port>63001</local-port>
          </tcp-params>
          <tls-params>
            <server-authentication>
              <ca-certs>explicitly-trusted-server-ca-certs</ca-certs>
              <server-certs>explicitly-trusted-server-certs</server-ce\
rts>
            </server-authentication>
          </tls-params>
        </channel>
        <channel>
          <tcp-params>
            <remote-address>192.0.2.1</remote-address>
            <remote-port>443</remote-port>
            <local-address>192.0.3.2</local-address>
            <local-port>63001</local-port>
          </tcp-params>
          <tls-params>
            <server-authentication>
```

```
               <ca-certs>explicitly-trusted-server-ca-certs</ca-certs>
               <server-certs>explicitly-trusted-server-certs</server-ce\
 rts>
             </server-authentication>
           </tls-params>
         </channel>
       </receiver>
   </receivers>

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

   <truststore xmlns="urn:ietf:params:xml:ns:yang:ietf-truststore">
     <certificates>
        <name>explicitly-trusted-server-certs</name>
        <description>
          Specific server authentication certificates for explicitly
          trusted servers.  These are needed for server certificates
          that are not signed by a pinned CA.
        </description>
        <certificate>
          <name>Fred Flintstone</name>
          <cert>base64encodedvalue==</cert>
        </certificate>
      </certificates>
      <certificates>
        <name>explicitly-trusted-server-ca-certs</name>
        <description>
          Trust anchors (i.e. CA certs) that are used to authenticate\
          server connections.  Servers are authenticated if their
          certificate has a chain of trust to one of these CA
          certificates.
```

```
        </description>
        <certificate>
          <name>ca.example.com</name>
          <cert>base64encodedvalue==</cert>
        </certificate>
      </certificates>
   </truststore>
 </config>
```

Appendix B. Change Log

(To be removed by RFC editor prior to publication)

v01

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

v02

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

v03

- o Added a section for Terminologies.
- o Added a section for Subscription State Change Notifications.
- o Improved the Publication Composition section by adding a method to check the integrity of the data generated from different Publishers at the same time period.
- o Revised the solution overview for a more clear description.

v04

- o Added the YANG data model for the proposed augment.

v05

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

v06

o Added examples.

Authors' Addresses

Tianran Zhou
Huawei
156 Beiqing Rd., Haidian District
Beijing
China

Email: zhoutianran@huawei.com

Guangying Zheng
Huawei
101 Yu-Hua-Tai Software Road
Nanjing, Jiangsu
China

Email: zhengguangying@huawei.com

Eric Voit
Cisco Systems
United States of America

Email: evoit@cisco.com

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

Email: ludwig@clemm.org

Andy Bierman
YumaWorks
United States of America

Email: andy@yumaworks.com