NETCONF
Internet-Draft
Intended status: Standards Track
Expires: January 8, 2020

T. Zhou
G. Zheng
Huawei
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

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

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

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

This Internet-Draft will expire on January 8, 2020.

Copyright Notice

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

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

Table of Contents

1. Introduction

Streaming telemetry refers to sending a continuous stream of operational data from a device to a remote receiver. This provides an ability to monitor a network from remote and to provide network analytics. Devices generate telemetry data and push that data to a

collector for further analysis. By streaming the data, much better performance, finer-grained sampling, monitoring accuracy, and bandwidth utilization can be achieved than with polling-based alternatives.

YANG-Push [I-D.ietf-netconf-yang-push] defines a transport-independent subscription mechanism for datastore updates, in which a subscriber can subscribe to a stream of datastore updates from a server, or update provider. The current design involves subscription to a single push server. This conceptually centralized model encounters efficiency limitations in cases where the data sources are themselves distributed, such as line cards in a piece of network equipment. In such cases, it will be a lot more efficient to have each data source (e.g., each line card) originate its own stream of updates, rather than requiring updates to be tunneled through a central server where they are combined. What is needed is a distributed mechanism that allows to directly push multiple individual data substreams, without needing to first pass them through an additional processing stage for internal consolidation, but still allowing those substreams to be managed and controlled via a single subscription.

This document will describe such distributed data collection mechanism and how it can work by extending existing YANG-Push mechanism. The proposal is general enough to fit many scenarios.

2. Use Cases

2.1. Use Case 1: Data Collection from Devices with Main-board and Line-cards

For data collection from devices with main-board and line-cards, existing YANG-Push solutions consider only one push server typically reside in the main board. As shown in the following figure, data are collected from line cards and aggregate to the main board as one consolidated stream. So the main board can easily become the performance bottle-neck. The optimization is to apply the distributed data collection mechanism which can directly push data from line cards to a collector. On one hand, this will reduce the cost of scarce compute and memory resources on the main board for data processing and assembling. On the other hand, distributed data push can off-load the streaming traffic to multiple interfaces.

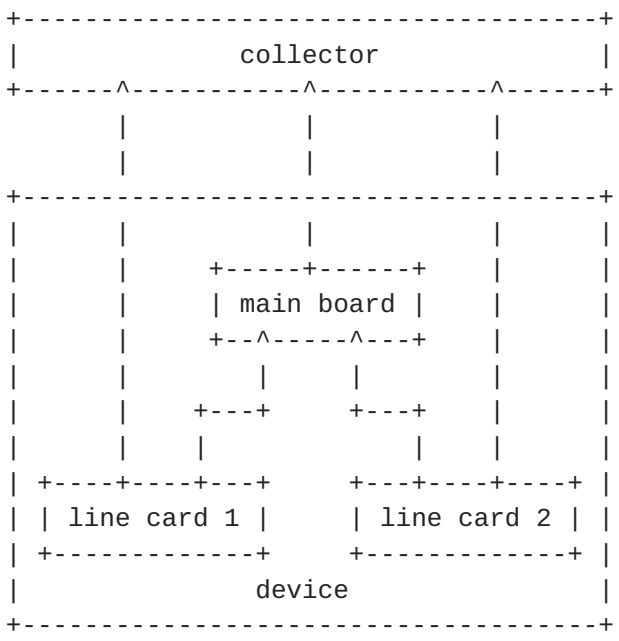

Fig. 1 Data Collection from Devices with Main-board and Line-cards

2.2. Use Case 2: IoT Data Collection

In the IoT data collection scenario, as shown in the following figure, collector usually cannot access to IoT nodes directly, but is isolated by the border router. So the collector subscribes data from the border router, and let the border router to disassemble the subscription to corresponding IoT nodes. The border router is typically the traffic convergence point. It's intuitive to treat the border router as a broker assembling the data collected from the IoT nodes and forwarding to the collector[I-D.ietf-core-coap-pubsub]. However, the border router is not so powerful on data assembling as a network device. It's more efficient for the collector, which may be a server or even a cluster, to assemble the subscribed data if possible. In this case, push servers that reside in IoT nodes can stream data to the collector directly while traffic only passes through the border router.

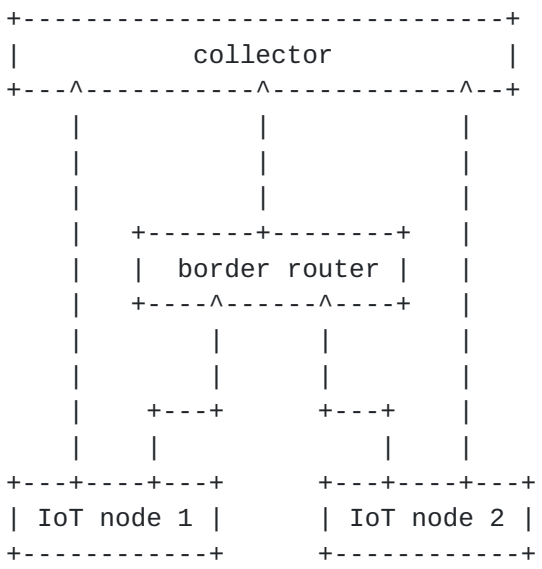

Fig. 2 IoT Data Collection

3. Terminologies

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

Receiver: is the target for the data publication.

Publisher: pushes data to the receiver according to the subscription information.

Subscription Server: which manages capabilities that it can provide to the subscriber.

Global Subscription: the subscription requested by the subscriber. It may be decomposed into multiple Component Subscriptions.

Component Subscription: is the subscription that defines the data from each individual telemetry source which is managed and controlled by a single Subscription Server.

Global Capability: is the overall subscription capability that the group of Publishers can expose to the Subscriber.

Component Capability: is the subscription capability that each Publisher can expose to the Subscriber.

Master Publication Channel: the session between the Master Publisher and the Receiver.

Agent Publication Channel: the session between the Agent Publisher and the Receiver.

4. Solution Overview

All the use cases described in the previous section are very similar on the data subscription and publication mode, hence can be abstracted to the following generic distributed data collection framework, as shown in the following figure.

A Collector usually includes two components,

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

- the Receiver is the target for the data publication.

For one subscription, there may be one to many receivers. And the subscriber does not necessarily share the same address with the receivers.

In this framework, the Publisher pushes data to the receiver according to the subscription information. The Publisher has the Master role and the Agent role. Both the Master and the Agent include the Subscription Server which actually manages capabilities that it can provide to the subscriber.

The Master knows all the capabilities that the attached Agents and itself can provide, and exposes the Global Capability to the Collector. The Collector cannot see the Agents directly, so it will only send the Global Subscription information to the Master. The Master disassembles the Global Subscription to multiple Component Subscriptions, each involving data from a separate telemetry source. The Component Subscriptions are then distributed to the corresponding Agents.

When data streaming, the Publisher collects and encapsulates the packets per the Component Subscription, and pushes the piece of data which can serve directly to the designated data Collector. The Collector is able to assemble many pieces of data associated with one Global Subscription, and can also deduce the missing pieces of data.

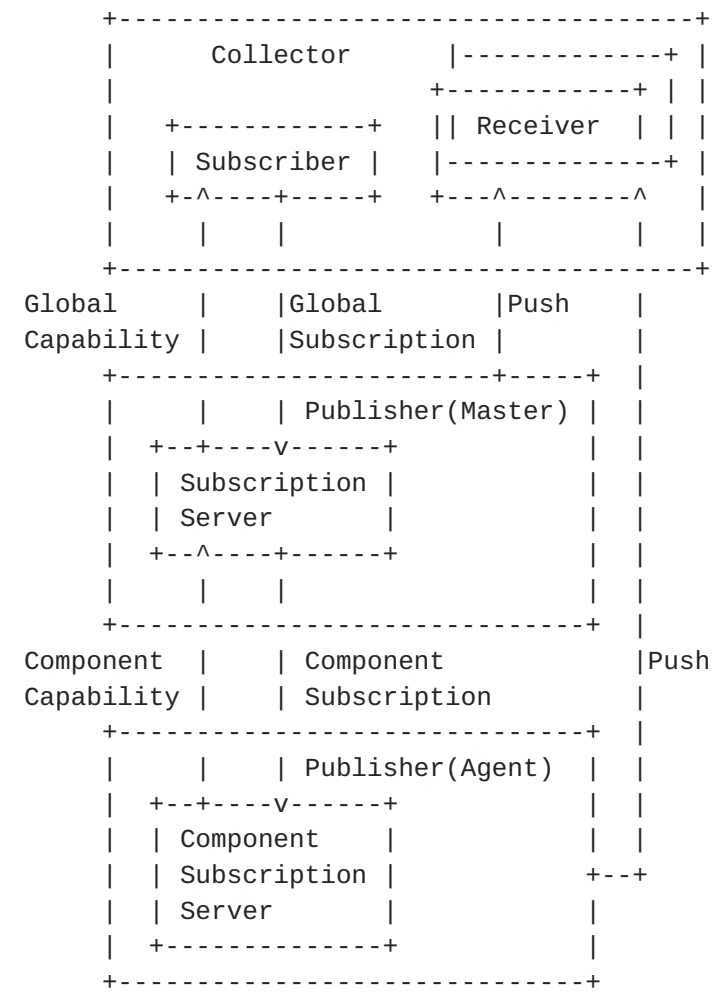

Fig. 3 The Generic Distributed Data Collection Framework

Master and Agents may interact with each other in several ways:

- Agents need to have a registration or announcement handshake with the Master, so the Master is aware of them and of life-cycle events (such as Agent appearing and disappearing).

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

- The Master relays the component subscriptions to the Agents.

- The Agents indicate status of Component Subscriptions to the Master. The status of the overall subscription is maintained by the Master. The Master is also responsible for notifying the subscriber in case of any problems of Component Subscriptions.

Any technical mechanisms or protocols used for the coordination of operational information between Master and Agent is out-of-scope of the solution. We will need to instrument the results of this coordination on the Master Node.

5. Subscription Decomposition

Since Agents are invisible to the Collector, the Collector can only subscribe to the Master. This requires the Master to:

1. expose the Global Capability that can be served by multiple Publishers;

2. disassemble the Global Subscription to multiple Component Subscriptions, and distribute them to the corresponding telemetry sources;

3. notify on changes when portions of a subscription moving between different Agents over time.

To achieve the above requirements, the Master need a Global Capability description which is typically the YANG [RFC7950] data model. This global YANG model is provided as the contract between the Master and the Collector. Each Agent associating with the Master owns a local YANG model to describe the Component Capabilities which it can serve as part of the Global Capability. All the Agents need to know the namespace associated with the Master.

The Master also need a data structure, typically a Resource-Location Table, to keep track of the mapping between the resource and the corresponding location of the Subscription Server which commits to serve the data. When a Global Subscription request arrives, the Master will firstly extract the filter information from the request. Consequently, according to the Resource-Location Table, the Global Subscription can be disassembled into multiple Component Subscriptions, and the corresponding location can be associated.

The decision whether to decompose a Global Subscription into multiple Component Subscriptions rests with the Resource-Location Table. A Master can decide to not decompose a Global Subscription at all and push a single stream to the receiver, because the location information indicates the Global Subscription can be served locally by the Master. Similarly, it can decide to entirely decompose a Global Subscription into multiple Component Subscriptions that each push their own streams, but not from the Master. It can also decide to decompose the Global Subscription into several Component Subscriptions and retain some aspects of the Global Subscription itself, also pushing its own stream.

Component Subscriptions belonging to the same Global Subscription MUST NOT overlap. The combination of all Component Subscriptions MUST cover the same range of nodes as the Global Subscription. Also, the same subscription settings apply to each Component Subscription, i.e., the same receivers, the same time periods, the same encodings are applied to each Component Subscription per the settings of the Global Subscription.

Each Component Subscription in effect constitutes a full-fledged subscription, with the following constraints:

- Component subscriptions are system-controlled, i.e. managed by the Master, not by the subscriber.

- Component subscription settings such as time periods, dampening periods, encodings, receivers adopt the settings of their Global Subscription.

- The life-cycle of the Component Subscription is tied to the life-cycle of the Global Subscription. Specifically, terminating/ removing the Global Subscription results in termination/removal of Component Subscriptions.

- The Component Subscriptions share the same Subscription ID as the Global Subscription.

6. Publication Composition

The Publisher collects data and encapsulates the packets per the Component Subscription. There are several potential encodings, including XML, JSON, CBOR and GPB. The format and structure of the data records are defined by the YANG schema, so that the composition at the Receiver can benefit from the structured and hierarchical data instance.

The Receiver is able to assemble many pieces of data associated with one subscription, and can also deduce the missing pieces of data. The Receiver recognizes data records associated with one subscription according the Subscription ID. Data records generated per one subscription are assigned with the same Subscription ID.

For the time series data stream, records are produced periodically from each stream originator. The message arrival time varies because of the distributed nature of the publication. The Receiver assembles data generated at the same time period based on the recording time consisted in each data record. In this case, time synchronization is required for all the Publishers.

To check the integrity of the data generated from different Publishers at the same time period, the Message Generator ID [I-D.ietf-netconf-notification-messages]is helpful. This requires the Subscriber to know the number of Component Subscriptions which the Global Subscription is decomposed to. For the dynamic subscription, the output of the "establish-subscription" and "modify-subscription" RPC defined in [I-D.ietf-netconf-subscribed-notifications] MUST include a list of Message Generator IDs to indicate how the Global Subscription is decomposed into several Component Subscriptions. The "subscription-started" and "subscription-modified" notification defined in [I-D.ietf-netconf-subscribed-notifications] MUST also include a list of Message Generator IDs to notify the current Publishers for the corresponding Global Subscription.

7. Subscription State Change Notifications

In addition to sending event records to receivers, the Master MUST also send subscription state change notifications[I-D.ietf-netconf-subscribed-notifications] when events related to subscription management have occurred. All the subscription state change notifications MUST be delivered by the Master Publication Channel which is the session between the Master Publisher and the Receiver.

When the subscription decomposition result changed, the "subscription-modified" notification MUST be sent to indicate the new list of Publishers.

8. YANG Tree

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

9. YANG Module

```
<CODE BEGINS> file "ietf-multiple-stream-originators@2019-07-07.yang"
module ietf-multiple-stream-originators {
  yang-version 1.1;
  namespace
    "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

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

v02

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

v03

- Added a section for Terminologies.
- Added a section for Subscription State Change Notifications.
- Improved the Publication Composition section by adding a method to check the integrity of the data generated from different Publishers at the same time period.
- Revised the solution overview for a more clear description.

v04

- Added the YANG data model for the proposed augment.

v05

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

v06

- Added examples.

Authors' Addresses

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

Email: zhoutianran@huawei.com

Guangying Zheng
Huawei
101 Yu-Hua-Tai Software Road
Nanjing, Jiangsu
China

Email: zhengguangying@huawei.com

Eric Voit
Cisco Systems
United States of America

Email: evoit@cisco.com

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

Email: ludwig@clemm.org

Andy Bierman
YumaWorks
United States of America

Email: andy@yumaworks.com