NETCONF
Internet-Draft
Intended status: Standards Track
Expires: May 7, 2020

T. Zhou
G. Zheng
Huawei
E. Voit
Cisco Systems
A. Clemm
Futurewai

November 4, 2019

Subscription to Multiple Stream Originators
draft-zhou-netconf-multi-stream-originators-10

Abstract

This document describes the distributed data export mechanism that allows multiple data streams to be managed by using a single subscription. Specifically, device can decide to decompse one subscription into multiple subscriptions to the line-cards. So that each line-card can directly push data to the collector without passing through a broker for internal consolidation. And the device can indicate the subscription decomposition result to the receiver to check the data integrity.

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 May 7, 2020.

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.

Mechanisms to support subscription to event notifications have been defined in[RFC8639] and [RFC8641]. 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 export mechanism and how it can work by extending existing push mechanism. Specifically, device can decide to decompse one subscription into multiple subscriptions to the line-cards. So that each line-card can directly push data to the collector without passing through a broker for internal consolidation. And the device can indicate the subscription decomposition result to the receiver to check the data integrity. The proposal will focus on the scenario when data collection from devices with main-board and line-cards. It could be generalized to other distributed data export scenarios.

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

For data collection from devices with main-board and line-cards, existing 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 export 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.

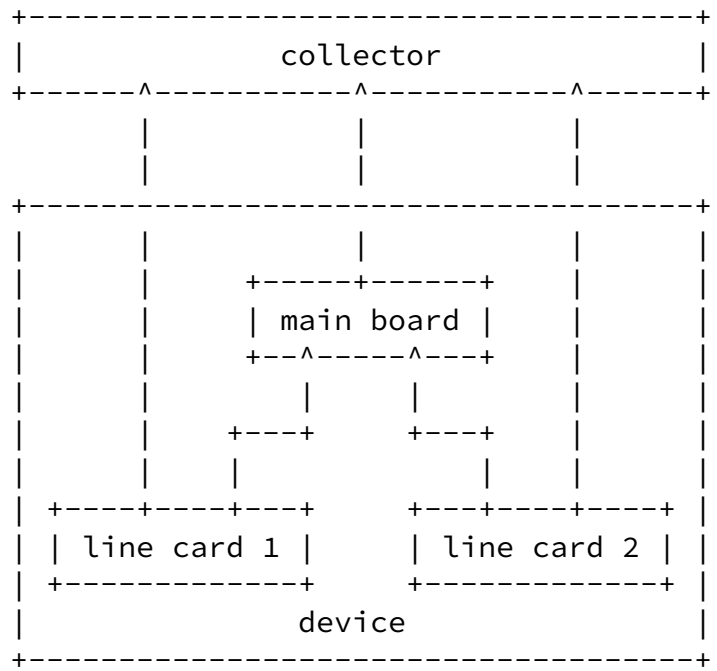


Fig. 1 Data Collection from Devices with Main-board and Line-cards

3. Terminologies

The following terms are defined in [RFC8639] and are not redefined here:

Subscriber

Publisher

Receiver

Subscription

In addition, this document defines the following terms:

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 Publisher.

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: is the Publisher that interacts with the Subscriber to deal with the Global Subscription. It decomposes the Global Subscription to multiple Component Subscriptions and interacts with the Agents.

Agent: is the Publisher that interacts with the Master to deal with the Component Subscription.

4. Solution Overview

Figure 2 below shows the distributed data export framework.

A collector usually includes two components,

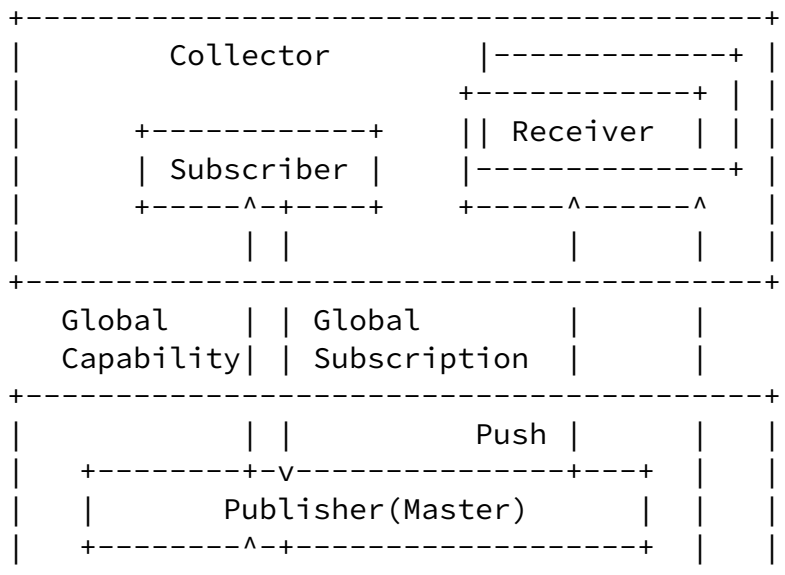
- the Subscriber generates the subscription instructions to express what and how the collector want to receive the data;

- 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. 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.



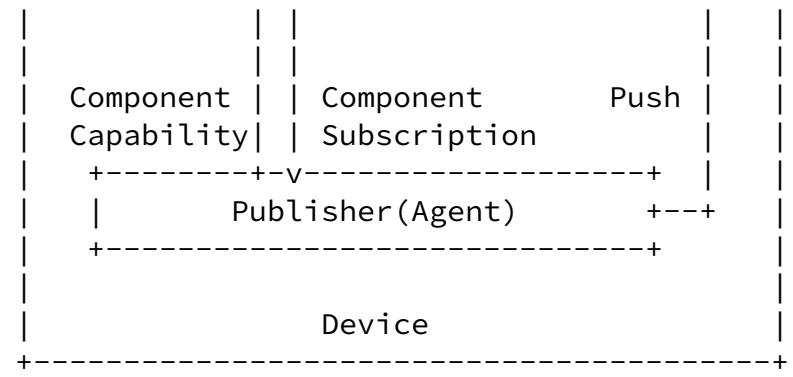


Fig. 2 The Distributed Data Export Framework

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

this document. We will need to instrument the results of this coordination on the Master.

5. Subscription Decomposition

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.

The Master can keep track of the mapping between the resource and the corresponding location of the Publisher which commits to serve the data. If a Publisher does not have external connectivity or permission to export, the master MUST NOT decompose a component subscription to that Publisher. In some implementations, the Global Subscription can be disassembled into multiple Component Subscriptions according to the Resource-Location Table, 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.

- 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 [RFC8639]. Data records generated per one subscription are assigned with the same Subscription ID.

For the periodic updates, records are produced periodically from each Publisher. 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.

Message Generator ID [I-D.ietf-netconf-notification-messages] is the identifier for the process which created the notification message. It's contained in every notification messages, and allows disambiguation of different line cards sending the messages. This document, in addition, requires the device to notify collector the set of Message Generator IDs standing for the Publishers serving for one Global Subscription. So that the collector can easily check the integrity of the data collected from different Publishers at the same time period. And the collector can deduce the Publishers which are responsible for the missing pieces of data.

For the dynamic subscription, the output of the "establish-subscription" RPC defined in [RFC8639] 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 [RFC8639] 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 [RFC8639] when events related to subscription management have occurred. All the subscription state change notifications MUST be delivered by the Master.

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

8. Publisher Configurations

This document assumes all the Publishers are preconfigured to be able to push data. The actual working Publishers are selected dynamically based on the subscription decomposition result. For UDP Publishers, the virtual IP address could be assigned for the publication. So all the UDP Publishers on the device can use the same source IP address configured, which may even not routeable. For connection based Publishers, e.g., HTTPS-based transport [I-D.ietf-netconf-https-notif], each Publisher MUST be able to receive packets from the receivers. This document does not restrict the way how the Publishers are accessible.

The specific configuration on transports is out of the scope of this document.

9. YANG Tree

```
   module: ietf-multiple-stream-originators
     augment /sn:subscriptions/sn:subscription:
       +--ro message-generator-id*   string
     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
```

10. YANG Module

```
 <CODE BEGINS> file "ietf-multiple-stream-originators@2019-10-12.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;
   }

   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
```

This version of this YANG module is part of RFC XXXX; see the RFC


```
    itself for full legal notices.";

  revision 2019-10-12 {
    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.";
    }
  }

  augment "/sn:subscriptions/sn:subscription" {
    description
      "This augmentation allows the message generators to be exposed
       for a subscription.";

    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 message-generator-ids;
  }
}
<CODE ENDS>
```

11. 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

12. Transport Considerations

The distributed data export mechanism enabled by this draft is expected to generate more data than YANG-Push [RFC8641]. 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.

13. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer

is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC5246].

The NETCONF Access Control Model (NACM) [RFC6536] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

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

- /subscriptions/subscription/message-generator-ids

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 [RFC8641].

14. Acknowledgements

We thank Kent Watsen, Mahesh Jethanandani, Martin Bjorklund, Tim Carey and Qin Wu for their constructive suggestions for improving this document.

15. References

15.1. 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>.

[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>.

[RFC8639] Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Subscription to YANG Notifications", RFC 8639, DOI 10.17487/RFC8639, September 2019, <https://www.rfc-editor.org/info/rfc8639>.

[RFC8641] Clemm, A. and E. Voit, "Subscription to YANG Notifications for Datastore Updates", RFC 8641, DOI 10.17487/RFC8641, September 2019, <https://www.rfc-editor.org/info/rfc8641>.

15.2. Informative References

[I-D.ietf-netconf-https-notif]
Jethanandani, M. and K. Watsen, "An HTTPS-based Transport for Configured Subscriptions", draft-ietf-netconf-https-notif-01 (work in progress), October 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-07 (work in progress), August 2019.

Appendix A. Examples

This appendix is non-normative.

A.1. Dynamic Subscription

Figure 3 shows a typical dynamic subscription to the device with distributed data export capability.

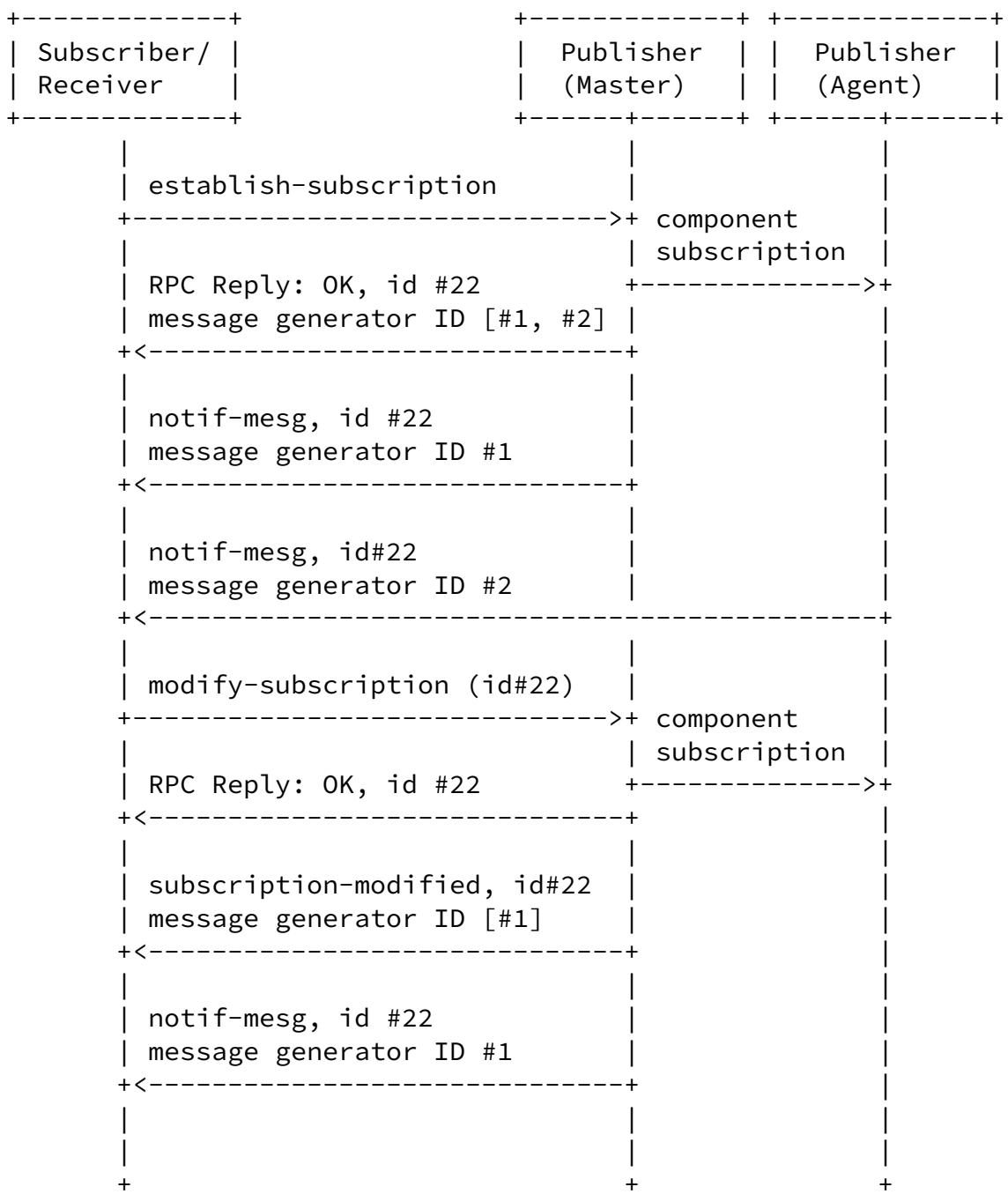


Fig. 3 Call Flow for Dynamic Subscription

A "establish-subscription" RPC request as per [RFC8641] is sent to the Master with a successful response. An example of using NETCONF 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="https://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>
```

Fig. 4 "establish-subscription" Request

As the device is able to fully satisfy the request, the request is given a subscription ID of 22. The response as in Figure 5 indicates that the subscription is decomposed into two component subscriptions which will be published by two message generators: #1 and #2.

```
<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">
      22
   </id>
   <message-generator-id
    xmlns="urn:ietf:params:xml:ns:yang:ietf-multiple-stream-originators>
      1
   </message-generator-id>
   <message-generator-id
    xmlns="urn:ietf:params:xml:ns:yang:ietf-multiple-stream-originators>
      2
   </message-generator-id>
</rpc-reply>
```

Fig. 5 "establish-subscription" Positive RPC Response

Then, both Publishers send notifications with the corresponding piece of data to the receiver.

The subscriber may invoke the "modify-subscription" RPC for a subscription it previously established. The RPC has no difference to the single publisher case as in [RFC8641]. Figure 6 provides 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>22</id>
    <yp:datastore
        xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
      ds:operational
    </yp:datastore>
    <yp:datastore-xpath-filter
        xmlns:ex="https://example.com/sample-data/1.0">
      /ex:bar
    </yp:datastore-xpath-filter>
    <yp:periodic>
      <yp:period>250</yp:period>
    </yp:periodic>
  </modify-subscription>
</rpc>
```

Fig. 6 "modify-subscription" Request

If the modification is successfully accepted, the "subscription-modified" subscription state notification is sent to the subscriber by the Master. The notification, Figure 7 for example, indicates the modified subscription is decomposed into one component subscription which will be published by message generator #1.

```
<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"
      xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-yang-push">
    <id>22</id>
    <yp:datastore
        xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
      ds:operational
    </yp:datastore>
    <yp:datastore-xpath-filter
        xmlns:ex="https://example.com/sample-data/1.0">
      /ex:bar
    </yp:datastore-xpath-filter>
    <yp:periodic>
        <yp:period>250</yp:period>
    </yp:periodic>
    <message-generator-id
    xmlns="urn:ietf:params:xml:ns:yang:ietf-multiple-stream-originators>
       1
    </message-generator-id>
  </subscription-modified>
</notification>
```

Fig. 7 "subscription-modified" Subscription State Notification

A.2. Configured Subscription

Figure 8 shows a typical configured subscription to the device with distributed data export capability.

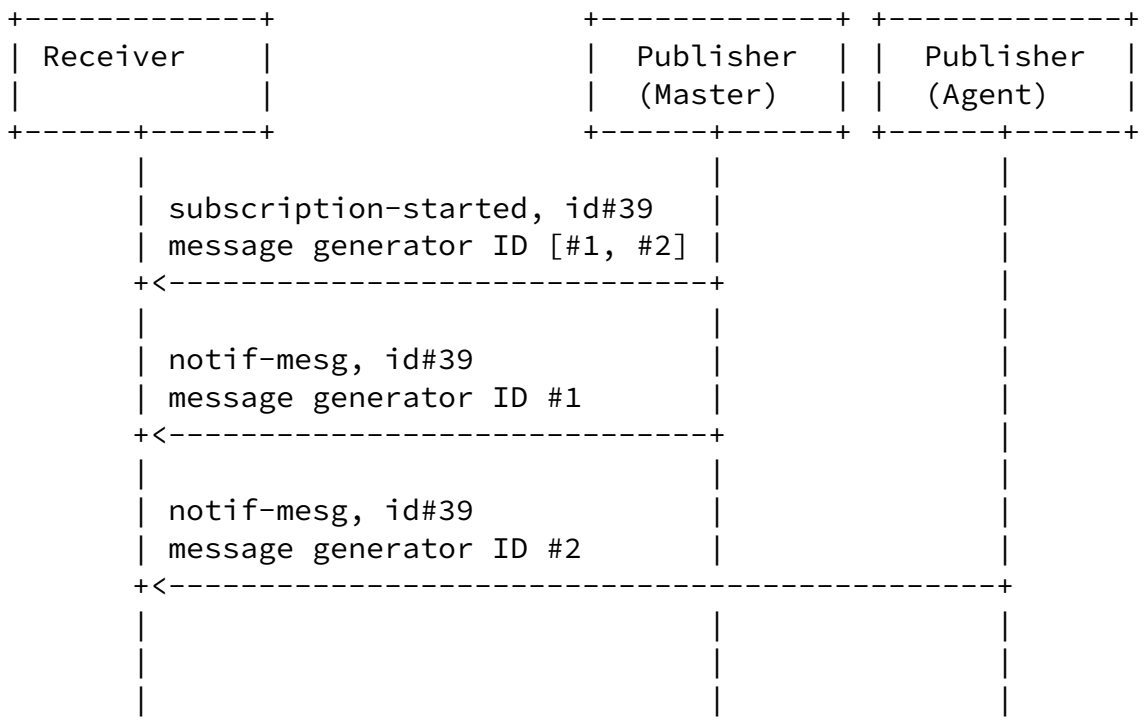


Fig. 8 Call Flow for Configured Subscription

Before starting to push data, the "subscription-started" subscription state notification is sent to the receiver. The following example assumes the NETCONF transport has already established. The notification indicates that the configured subscription is decomposed

into two component subscriptions which will be published by two message generators: #1 and #2.


```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2007-09-01T10:00:00Z</eventTime>
  <subscription-started
      xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications"
      xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-yang-push">
    <identifier>39</identifier>
    <yp:datastore
        xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
      ds:operational
    </yp:datastore>
    <yp:datastore-xpath-filter
        xmlns:ex="https://example.com/sample-data/1.0">
      /ex:foo
    </yp:datastore-xpath-filter>
    <yp:periodic>
        <yp:period>250</yp:period>
    </yp:periodic>
    <message-generator-id
    xmlns="urn:ietf:params:xml:ns:yang:ietf-multiple-stream-originators>
```

```
      1
    </message-generator-id>
    <message-generator-id
    xmlns="urn:ietf:params:xml:ns:yang:ietf-multiple-stream-originators>
      2
    </message-generator-id>
  </subscription-started>
</notification>
```

Fig. 9 "subscription-started" Subscription State Notification

Then, both Publishers send notifications with the corresponding piece of data to the receiver.

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.

v07

- Removed the IoT use case.
- Revised examples.
- Add discussion on Publisher Configurations in section 8.

v08

- Cleared up the document to fix Terms, nits.
- Reduced internal implementation descriptions.
- Clarified Message Generator ID.

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