

NETCONF
Internet-Draft
Intended status: Standards Track
Expires: April 5, 2018

A. Gonzalez Prieto
VMware
A. Clemm
Huawei
E. Voit
E. Nilsen-Nygaard
A. Tripathy
Cisco Systems
October 2, 2017

NETCONF Support for Event Notifications
draft-ietf-netconf-netconf-event-notifications-05

Abstract

This document defines how to transport network subscriptions and event messages on top of the Network Configuration protocol (NETCONF). This includes the full set of RPCs, subscription state changes, and subscribed content needing asynchronous delivery.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

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

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

This Internet-Draft will expire on April 5, 2018.

Copyright Notice

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

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

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

Table of Contents

1.	Introduction	2
2.	Terminology	2
3.	Solution	3
3.1.	Event Stream Discovery	3
3.2.	Mandatory NETCONF support	4
3.3.	Dynamic Subscriptions	4
3.4.	Configured Subscriptions	11
4.	Interleave Capability	23
5.	Security Considerations	24
6.	Acknowledgments	24
7.	References	25
7.1.	Normative References	25
7.2.	Informative References	25
Appendix A.	Open Items	26
Appendix B.	Changes between revisions	26
B.1.	v04 to v05	26
B.2.	v03 to v04	26
B.3.	v01 to v03	26
B.4.	v00 to v01	26
	Authors' Addresses	27

[1.](#) Introduction

This document defines mechanisms that provide a subscription and asynchronous message notification delivery service for the NETCONF protocol [[RFC6241](#)] based on [[subscribe](#)]. This is an optional capability built on top of the base NETCONF definition.

[2.](#) Terminology

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

The following terms are defined in [[RFC6241](#)]: client, server, operation, RPC.

The following terms are defined in [[subscribe](#)]: event, event notification, stream, publisher, receiver, subscriber, subscription, configured subscription.

Note that a publisher in [[subscribe](#)] corresponds to a server in [[RFC6241](#)]. Similarly, a subscriber corresponds to a client. A receiver is also a client. In the remainder of this document, we will use the terminology in [[RFC6241](#)].

[3.](#) Solution

In this section, we describe and exemplify how [[subscribe](#)] is to be supported over NETCONF.

[3.1.](#) Event Stream Discovery

In the context of [[subscribe](#)] 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 server using the "get" operation against the top-level container "/streams" defined in [[subscribe](#)]. The reply includes the list of streams supported on the NETCONF server.

The following example illustrates the retrieval of the list of available event streams using the <get> operation.

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

Figure 1: Get streams request

The NETCONF server returns a list of event streams available. In this example, the list contains the NETCONF, SNMP, and SYSLOG streams.


```
<rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <streams
      xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
      <stream>
        <stream>NETCONF</stream>
        <description>The NETCONF mandatory event stream</description>
      </stream>
      <stream>
        <stream>SNMP</stream>
        <description>The SNMP event stream</description>
      </stream>
      <stream>
        <stream>SYSLOG</stream>
        <description>The SYSLOG event stream</description>
        <replay-support/>
        <replay-log-creation-time>
          2015-03-09T19:14:55.233Z</replay-log-creation-time>
        </stream>
      </streams>
    </data>
  </rpc-reply>
```

Figure 2: Get streams response

For [[yang-push](#)], a similar get is needed to retrieve available datastore names.

3.2. Mandatory NETCONF support

A NETCONF server implementation supporting [[subscribe](#)] must support dynamic subscriptions and the "NETCONF" notification event stream. The NETCONF event stream contains all NETCONF XML event information supported by the server.

A NETCONF server implementation supporting [[yang-push](#)] must support the "running" datastore.

3.3. Dynamic Subscriptions

3.3.1. Establishing Dynamic Subscriptions

The dynamic subscription RFC and interactions operation is defined in [[subscribe](#)].

3.3.1.1. Usage Example

An example of interactions over NETCONF transport for one sample subscription is below:

```
<netconf:rpc netconf:message-id="102"
  xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
  <establish-subscription
    xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
    <stream>NETCONF</stream>
    <event-filter-type>xpath-event-filter</event-filter-type>
    <event-filter-contents xmlns:ex="http://example.com/event/1.0"
      select="/ex:event[ex:eventClass='fault' and
        (ex:severity='minor' or ex:severity='major'
          or ex:severity='critical')]" />
    </establish-subscription>
  </netconf:rpc>
```

Figure 3: establish-subscription over NETCONF

3.3.1.2. Positive Response

If the NETCONF server can satisfy the request, the server sends a positive <subscription-result> element, and the subscription-id of the accepted subscription.

```
<rpc-reply message-id="102"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <subscription-result
    xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
    ok
  </subscription-result>
  <identifier
    xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
    52
  </identifier>
</rpc-reply>
```

Figure 4: Successful establish-subscription

3.3.1.3. Negative Response

If the NETCONF server cannot satisfy the request, or the client has no authorization to establish the subscription, the server will send a negative <subscription-result> element. For instance:


```
<rpc-reply message-id="103"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <subscription-result
    xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
    stream-unavailable
  </subscription-result>
</rpc-reply>
```

Figure 5: Unsuccessful establish subscription

3.3.1.4. Subscription Negotiation

If the client request includes parameters the NETCONF server cannot serve, the negative <subscription-result> may include hints at subscription parameters which would have been accepted. For instance, consider the following subscription from [[yang-push](#)], which augments the establish-subscription with some additional parameters, including "period". If the client requests a period which the NETCONF server cannot serve, the back-and-forth exchange may be:

```
<netconf:rpc message-id="101"
  xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
  <establish-subscription
    xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
    <datastore>running</datastore>
    <event-filter-type>
      subtree</event-filter-type>
    <event-filter-contents
      xmlns:ex="http://example.com/sample-data/1.0">
        select="/ex:foo" />
    <dampening-period>10</dampening-period>
    <encoding>encode-xml</encoding>
  </establish-subscription>
</netconf:rpc>

<rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <subscription-result
    xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
    error-insufficient-resources
  </subscription-result>
  <period-hint
    xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
    2000
  </period-hint>
</rpc-reply>
```

Figure 6: Subscription establishment negotiation

3.3.1.5. Message Flow Examples

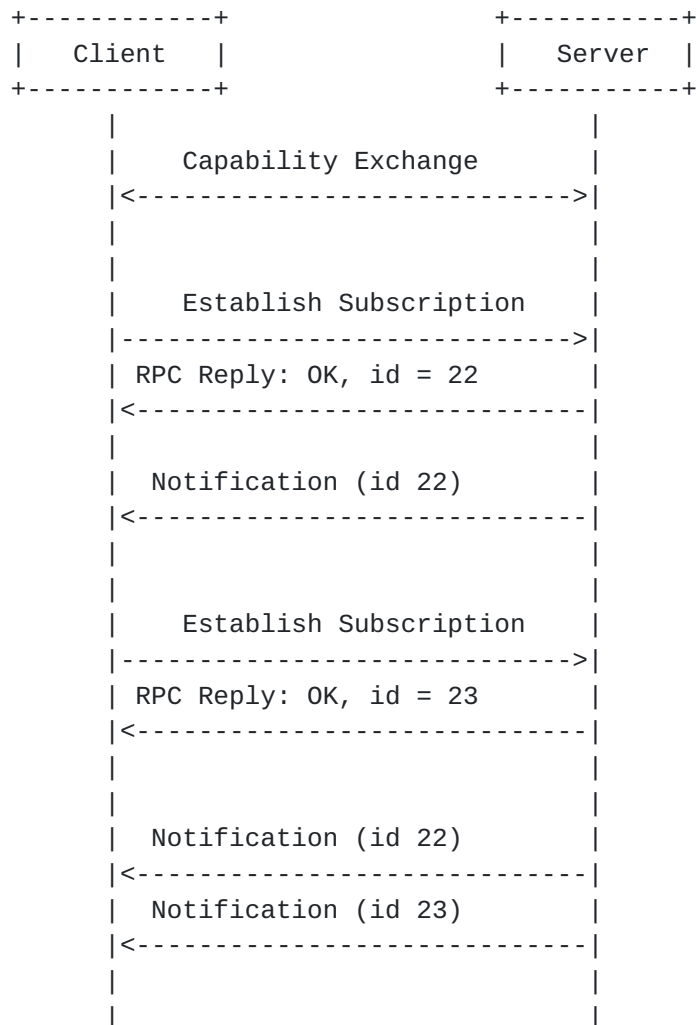


Figure 7: Multiple subscription establishments over a single NETCONF session

3.3.2. Modifying a Subscription

This operation is defined in [[subscribe](#)].

3.3.2.1. Usage Example

The following demonstrates modifying a subscription. Consider a subscription from [[yang-push](#)], which augments the establish-subscription with some additional parameters, including "period". A subscription may be established and modified as follows.


```
<netconf:rpc message-id="101"
  xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
  <establish-subscription
    xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
    <datastore xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">
      running</datastore>
    <event-filter-type>subtree-event-filter</event-filter-type>
    <event-filter-contents
      xmlns:ex="http://example.com/sample-data/1.0">
      select="/ex:foo" />
    <period xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
      1000
    </period>
  </establish-subscription>
</netconf:rpc>

<rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <subscription-result
    xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
    ok
  </subscription-result>
  <identifier
    xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
    1922
  </identifier>
</rpc-reply>

<netconf:rpc message-id="102"
  xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
  <modify-subscription
    xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
    <identifier>1922</identifier>
    <period>100</period>
  </modify-subscription>
</netconf:rpc>
```

Figure 8: Subscription modification

3.3.2.2. Positive Response

If the NETCONF server can satisfy the request, the server sends a positive `<subscription-result>` element. This response is like that to an `establish-subscription` request, but without the subscription identifier.


```
<rpc-reply message-id="102"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <subscription-result
    xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
    ok
  </subscription-result>
</rpc-reply>
```

Figure 9: Successful modify subscription

3.3.2.3. Negative Response

If the NETCONF server cannot satisfy the request, the server sends a negative `<subscription-result>` element. Its contents and semantics are identical to those in an establish-subscription request. For instance:

```
<rpc-reply message-id="102"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <subscription-result
    xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
    period-unsupported
  </subscription-result>
  <period-hint>500</period-hint>
</rpc-reply>
```

Figure 10: Unsuccessful modify subscription

3.3.2.4. Message Flow Example

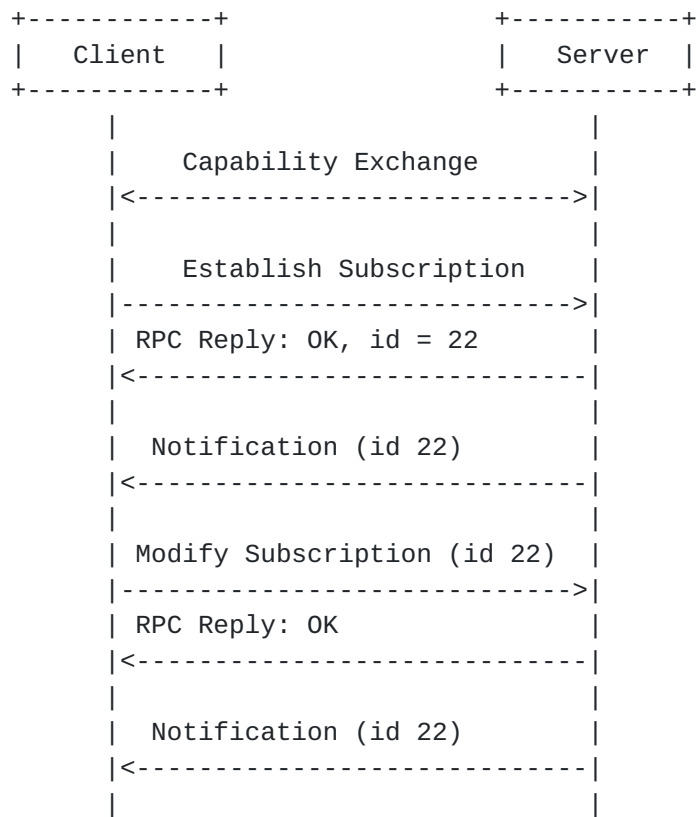


Figure 11: Message flow for successful subscription modification

3.3.3. Deleting a Subscription

This operation is defined in [[subscribe](#)] for events, and enhanced in [[yang-push](#)] for datastores.

3.3.3.1. Usage Example

The following demonstrates deleting a subscription.

```

<netconf:rpc message-id="101"
  xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
  <delete-subscription
    xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
    <identifier>1922</identifier>
  </delete-subscription>
</netconf:rpc>

```

Figure 12: Delete subscription

3.3.3.2. Positive Response

If the NETCONF server can satisfy the request, the server sends a positive `<subscription-result>` element. For example:

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

Figure 13: Successful delete subscription

3.3.3.3. Negative Response

If the NETCONF server cannot satisfy the request, the server sends an `<subscription-result>` element indicating the deletion was not performed. For example:

```
<rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <subscription-result
    xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
    no-such-subscription
  </subscription-result>
</rpc-reply>
```

Figure 14: Unsuccessful delete subscription

3.4. Configured Subscriptions

Configured subscriptions are established, modified, and deleted using configuration operations against the top-level subtree of [[subscribe](#)] or [[yang-push](#)] via configuration interface. This document covers configured subscriptions where the chosen protocol to send the notifications to the receivers is NETCONF.

Configured subscriptions are supported by NETCONF servers using NETCONF Call Home [[RFC8071](#)].

Any configuration interface can be used to set a configured subscription that uses NETCONF to push notifications to receivers. Without loss of generality, the examples in this section, use a NETCONF interface to configure the subscriptions

3.4.1. Establishing a Configured Subscription

For subscription establishment, a NETCONF client may send:

```
<rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <subscription-config
      xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
      <subscription>
        <identifier>
          1922
        </identifier>
        <stream>
          foo
        </stream>
        <receiver>
          <address>
            1.2.3.4
          </address>
          <port>
            1234
          </port>
          <protocol>
            netconf
          </protocol>
        </receiver>
      </subscription>
    </subscription-config>
  </edit-config>
</rpc>
```

Figure 15: Establish configured subscription

if the request is accepted, the server would reply:

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

Figure 16: Response to a successful configuration subscription establishment

if the request is not accepted because the server cannot serve it, no configuration is changed. In this case the server may reply:

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <subscription-result
    xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
    error-insufficient-resources
  </subscription-result>
</rpc-reply>
```

Figure 17: Response to a failed configured subscription establishment

For every configured receiver, once NETCONF transport session between the server and the receiver is recognized as active, the server will issue a "subscription-started" notification. After that, the server will send notifications to the receiver as per the subscription notification. Note that the server assumes that the receiver is ready to accept notifications on the NETCONF session. This may require coordination between the client that configures the subscription and the clients for which the notifications are intended. This coordination is out of the scope of this document.

The session is only intended for pushing notifications. Client requests on that session SHOULD be ignored by the server.

The contents sent by the server on the Call Home session, once established, are identical to those in a dynamic subscription.

3.4.2. Call Home for Configured Subscriptions

Once the subscription configuration is active, if NETCONF transport is needed but does not exist to one or more target IP address plus port, the server initiates a transport session via [[RFC8071](#)] to those receiver(s) in the subscription using the address and port specified.

3.4.3. Full Establish Message Flow

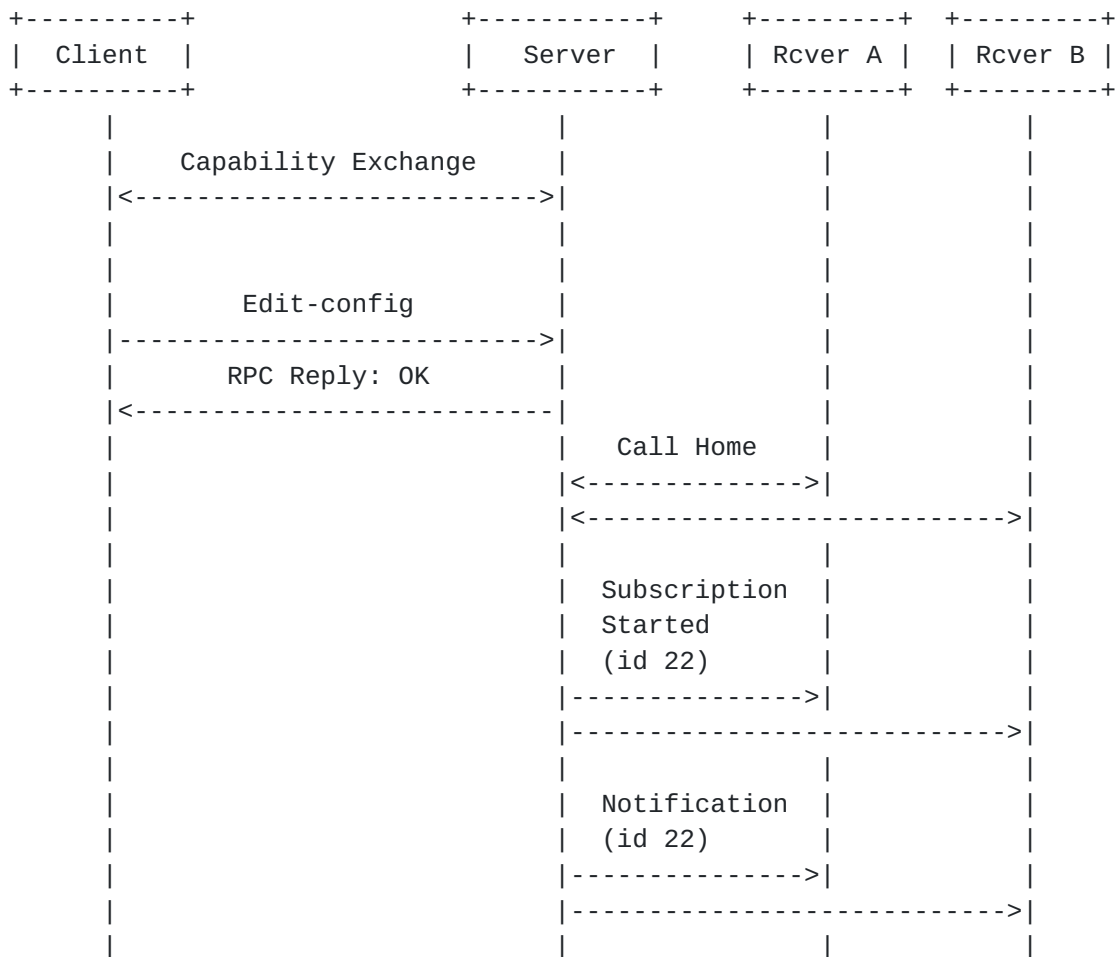


Figure 18: Message flow for configured subscription establishment

3.4.4. Modifying a Configured Subscription

Configured subscriptions can be modified using configuration operations against the top-level subtree `subscription-config`.

For example, the subscription established in the previous section could be modified as follows, choosing a different receiver:


```
<rpc message-id="102"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <subscription-config
      xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
      <subscription>
        <identifier>
          1922
        </identifier>
        <stream>
          foo
        </stream>
        <receiver>
          <address>
            1.2.3.5
          </address>
          <port>
            1234
          </port>
        </receiver>
      </subscription>
    </subscription-config>
  </edit-config>
</rpc>
```

Figure 19: Modify configured subscription

if the request is accepted, the server would reply:

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

Figure 20: A successful configured subscription modification

3.4.4.1. Message Flow Example

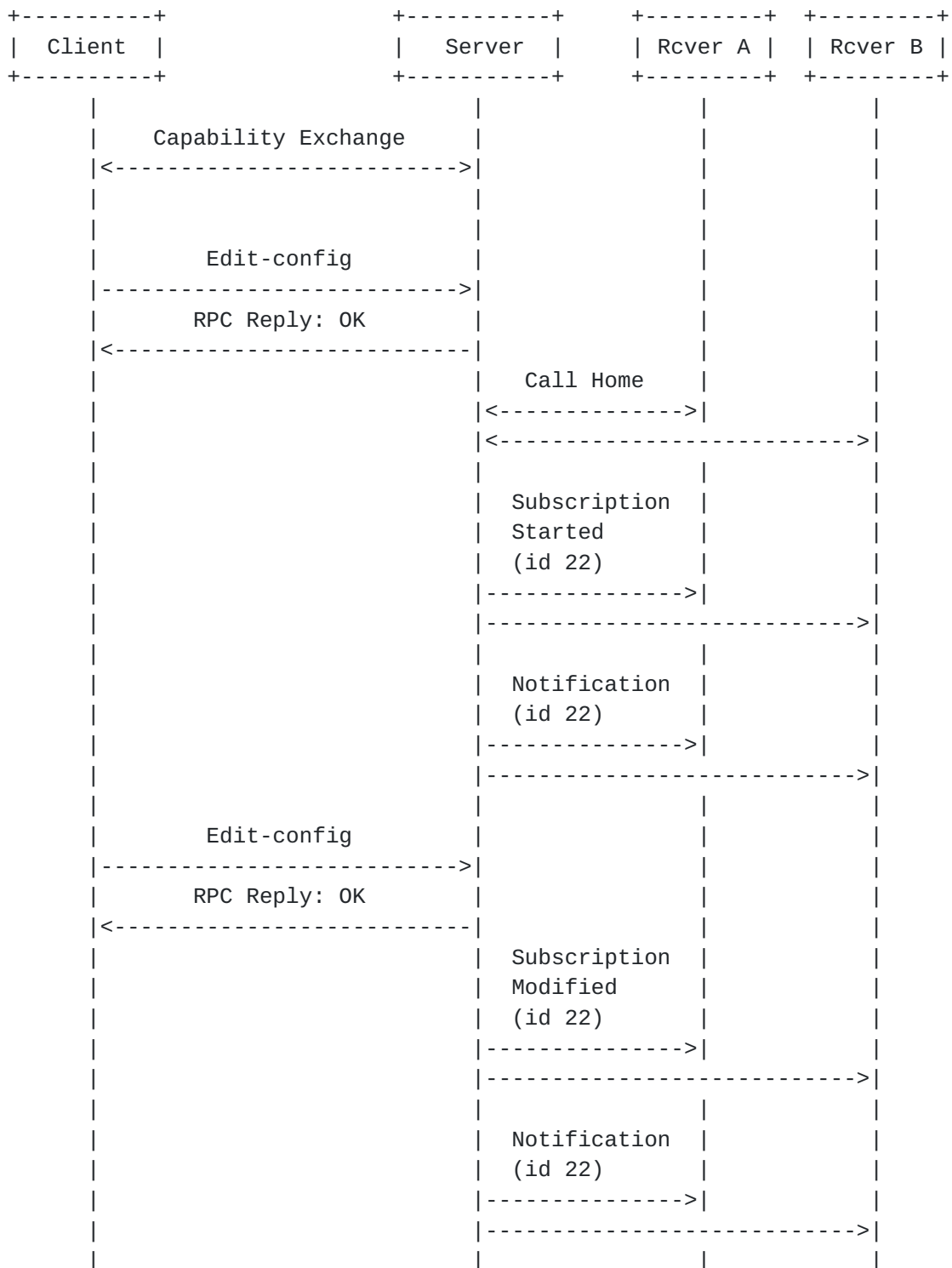


Figure 21: Message flow for subscription modification (configured subscription)

3.4.5. Deleting a Configured Subscription

Subscriptions can be deleted using configuration operations against the top-level subtree subscription-config. For example:

```
<rpc message-id="103"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <subscription-config
      xmlns:xc="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
      <subscription xc:operation="delete">
        <identifier>
          1922
        </identifier>
      </subscription>
    </subscription-config>
  </edit-config>
</rpc>

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

Figure 22: Deleting a configured subscription

3.4.5.1. Message Flow Example

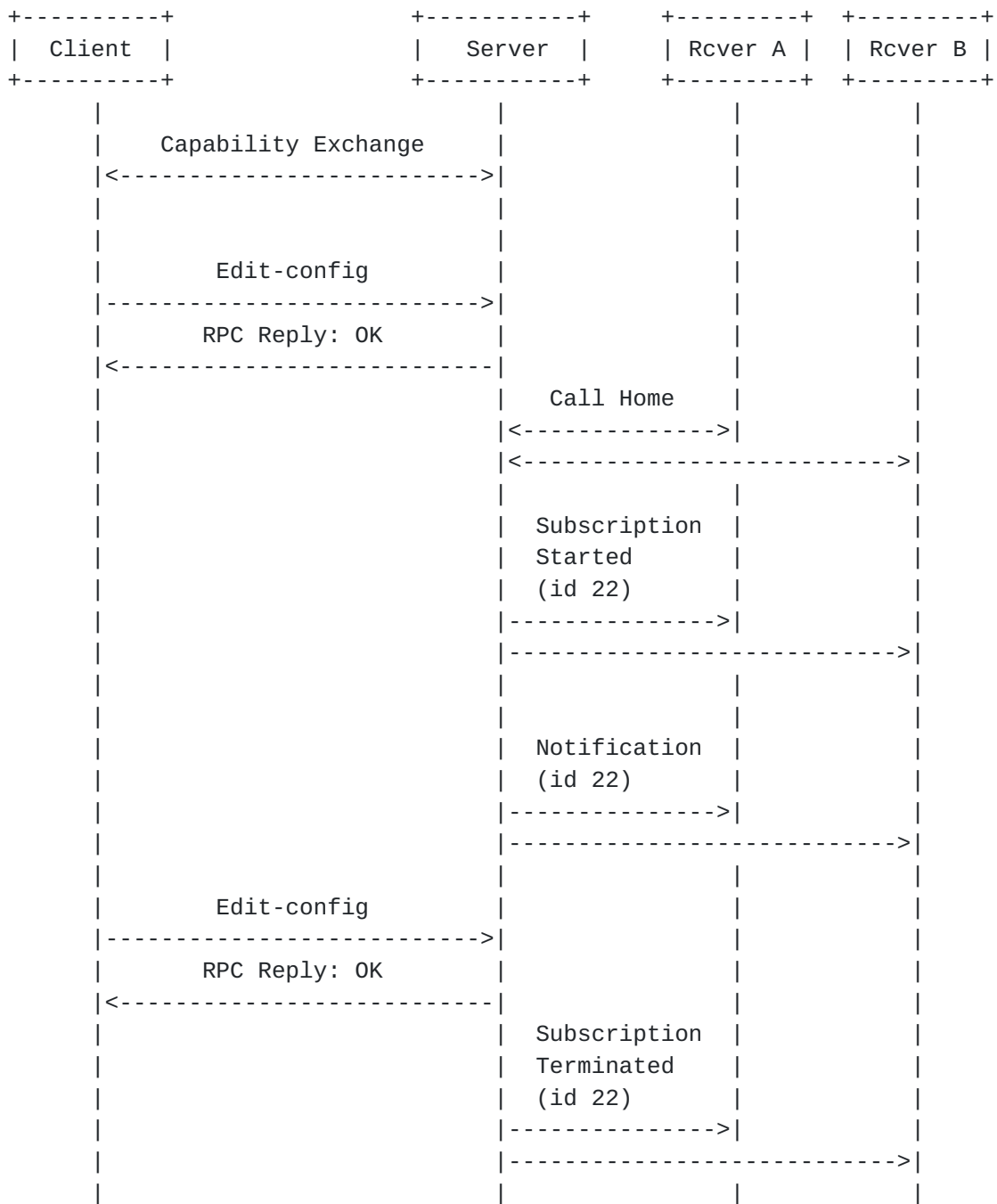


Figure 23: Message flow for subscription deletion (configured subscription)

3.4.6. Event (Data Plane) Notifications

Once a dynamic or configured subscription has been created, the NETCONF server sends (asynchronously) event notifications from the subscribed stream to receiver(s) over NETCONF. We refer to these as

data plane notifications. The data model for Event Notifications is defined in [[subscribe](#)].

The following is an example of an event notification from [[RFC6020](#)]:

```
notification link-failure {
  description "A link failure has been detected";
  leaf if-name {
    type leafref {
      path "/interface/name";
    }
  }
  leaf if-admin-status {
    type admin-status;
  }
  leaf if-oper-status {
    type oper-status;
  }
}
```

Figure 24: Definition of an event notification

This notification might result in the following, prior to it being placed into NETCONF. Note that the mandatory `eventTime` and `Subscription id` have been added.

```
<notification
  xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push: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 25: Event notification

[3.4.7.](#) Subscription State Notifications

In addition to data plane notifications, a publisher may send subscription state notifications (defined in [[subscribe](#)]) to indicate to receivers that an event related to the subscription management has occurred. Subscription state notifications cannot be filtered out. Next we exemplify them:

3.4.7.1. subscription-started and subscription-modified

A subscription-started would look like:

```
<notification
  xmlns=" urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
  <eventTime>2007-09-01T10:00:00Z</eventTime>
  <subscription-started
    xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0"/>
    <identifier>39</identifier>
    <stream>NETCONF</stream>
  </subscription-started/>
</notification>
```

Figure 26: subscription-started subscription state notification

The subscription-modified is identical, with just the word "started" being replaced by "modified".

3.4.7.2. subscription-completed, subscription-resumed, and replay-complete

A subscription-completed would look like:

```
<notification
  xmlns=" urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
  <eventTime>2007-09-01T10:00:00Z</eventTime>
  <subscription-completed
    xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
    <identifier>39</identifier>
  </subscription-completed>
</notification>
```

Figure 27: subscription-completed notification in XML

The equivalent using JSON encoding would be:


```
<notification
  xmlns=" urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
  <eventTime>2007-09-01T10:00:00Z</eventTime>
  <notification-contents-json>
    {
      "sn:subscription-completed": {
        "identifier" : 39
      }
    }
  </notification-contents-json>
</notification>
```

Figure 28: 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" in both encodings.

3.4.7.3. subscription-terminated and subscription-suspended

A subscription-terminated would look like:

```
<notification
  xmlns=" urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
  <eventTime>2007-09-01T10:00:00Z</eventTime>
  <subscription-terminated
    xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
    <identifier>39</identifier>
    <error-id>error-insufficient-resources</error-id>
  </subscription-terminated>
</notification>
```

Figure 29: subscription-modified subscription state notification

The above, and the subscription-suspended are virtually identical, with "subscription-terminated" simply being replaced by "subscription-suspended".

3.4.7.4. Notification Message Flow Examples

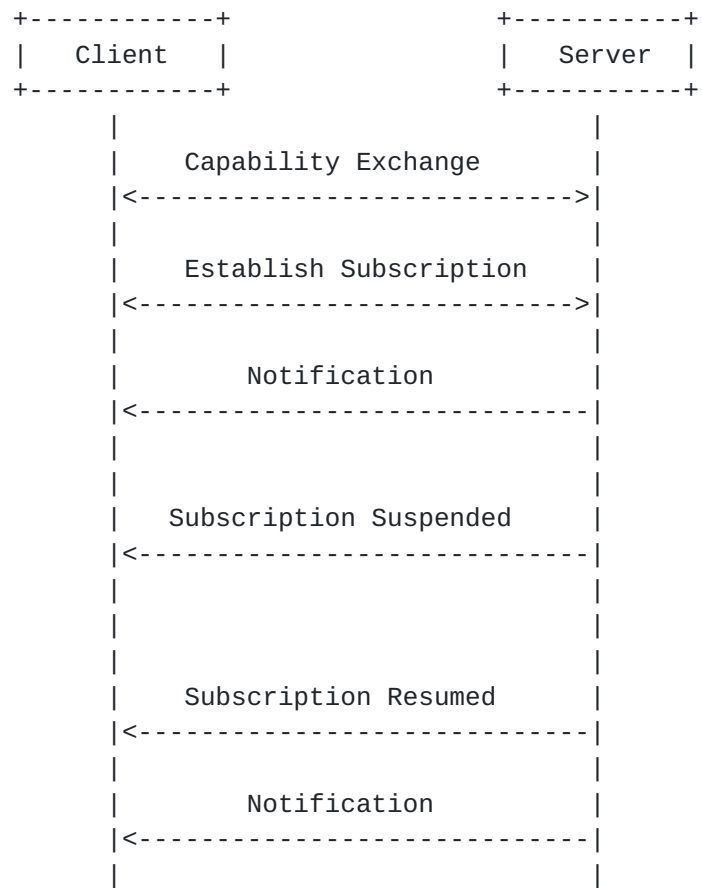


Figure 30: subscription-suspended and resumed notifications

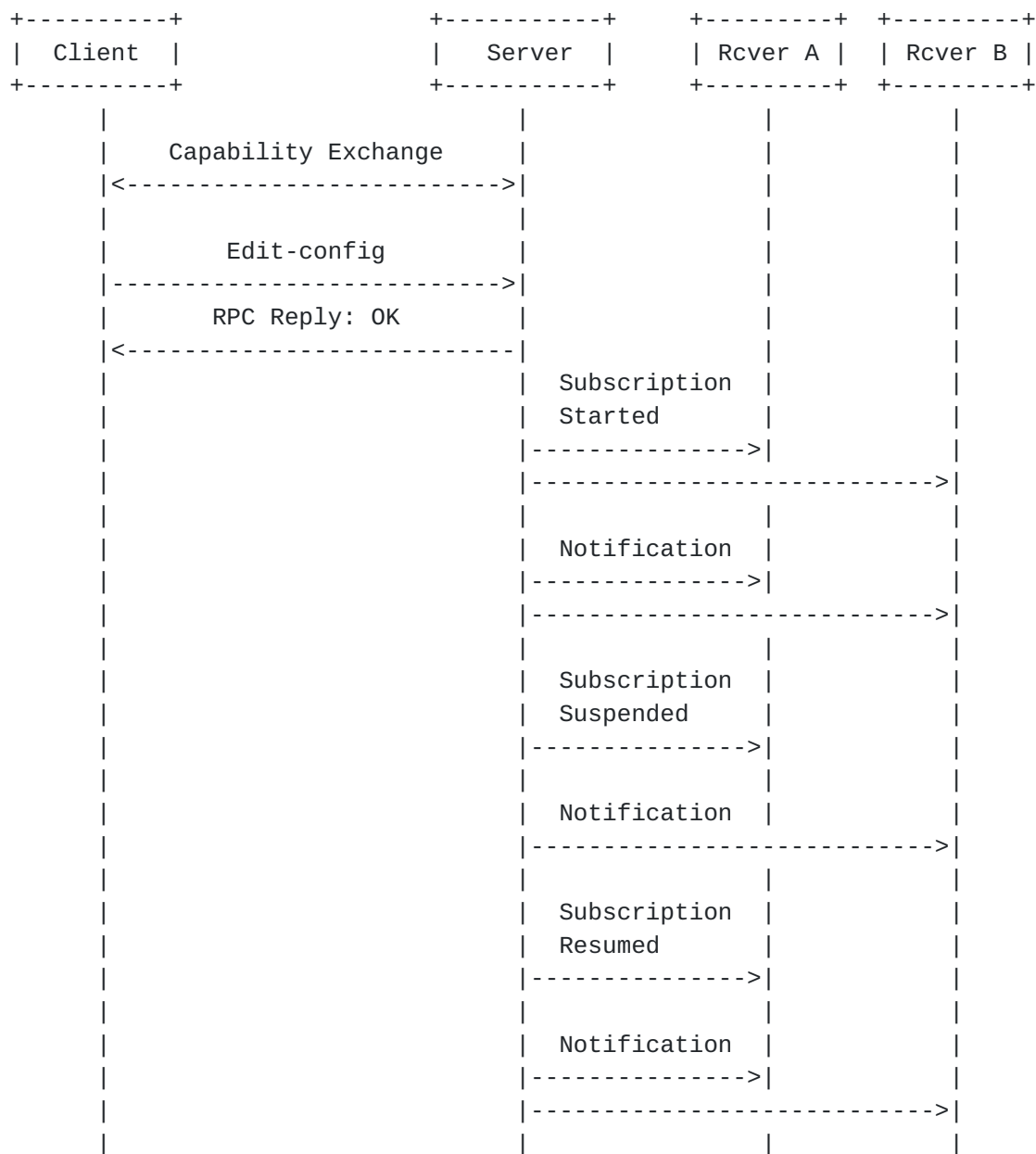


Figure 31: Suspended and resumed notifications for a single configured receiver

4. Interleave Capability

The `:interleave` capability is originally defined in [\[RFC5277\]](#). It is incorporated in this document with essentially the same semantics. That is, the NETCONF server MUST receive, process, and respond to NETCONF requests on a session with active notification subscriptions.

The `:interleave` capability is identified by the following string:
`urn:ietf:params:netconf:capability:interleave:1.0`

Note that subscription operations **MUST** be received, processed, and responded on a session with active notification subscriptions. That mandatory requirement together with the `:interleave` capability permits a client performing all operations against a server using a single connection, allowing for better scalability with respect to the number of NETCONF sessions required to manage an entity.

5. Security Considerations

The `<notification>` elements are never sent before the transport layer and the NETCONF layer, including capabilities exchange, have been established and the manager has been identified and authenticated.

A secure transport must be used and the server must ensure that the user has sufficient authorization to perform the function they are requesting against the specific subset of content involved.

The contents of notifications, as well as the names of event streams, may contain sensitive information and care should be taken to ensure that they are viewed only by authorized users. The NETCONF server **MUST NOT** include any content in a notification that the user is not authorized to view.

If a malicious or buggy NETCONF client sends a number of `<establish-subscription>` requests, then these subscriptions accumulate and may use up system resources. In such a situation, subscriptions **MAY** be terminated by terminating the suspect underlying NETCONF sessions. The server **MAY** also suspend or terminate a subset of the active subscriptions on the NETCONF session.

Configured subscriptions from one or more publishers could be used to overwhelm a receiver, which perhaps doesn't even support subscriptions. Clients that do not want pushed data need only terminate or refuse any transport sessions from the publisher.

The NETCONF Authorization Control Model [[RFC6536](#)] **SHOULD** be used to control and restrict authorization of subscription configuration. This control model permits specifying per-user permissions to receive specific event notification types. The permissions are specified as a set of access control rules.

6. 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, Kent Watsen, 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>>.
- [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>>.
- [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>>.
- [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>>.

7.2. Informative References

- [subscribe] Voit, Eric., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Custom Subscription to Event Notifications", September 2017, <<https://datatracker.ietf.org/doc/draft-ietf-netconf-subscribed-notifications/>>.
- [yang-push] Clemm, A., Gonzalez Prieto, A., Voit, Eric., Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "Subscribing to YANG datastore push updates", September 2017, <<https://datatracker.ietf.org/doc/draft-ietf-netconf-yang-push/>>.

Appendix A. Open Items

(To be removed by RFC editor prior to publication)

- o Formal definition of: notification-contents-json, and incorporation of the subscription-id in the notifications. It depends on the formal definition of the notification element

Appendix B. Changes between revisions

(To be removed by RFC editor prior to publication)

B.1. v04 to v05

- o Text presentation modifications throughout
- o Modified examples to match the namespace, prefixes, and data node identifiers in [[subscribe](#)] and [[yang-push](#)]
- o Modified examples to include <subscription-result> in all RPC responses
- o Modified examples to include mandatory fields in [[subscribe](#)] and [[yang-push](#)]

B.2. 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.3. v01 to v03

- o Text simplifications throughout
- o v02 had no meaningful changes

B.4. 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

Alberto Gonzalez Prieto
VMware

Email: agonzalezpri@vmware.com

Alexander Clemm
Huawei

Email: ludwig@clemm.org

Eric Voit
Cisco Systems

Email: evoit@cisco.com

Einar Nilsen-Nygaard
Cisco Systems

Email: einarnn@cisco.com

Ambika Prasad Tripathy
Cisco Systems

Email: ambtripa@cisco.com

