

NETCONF
Internet-Draft
Intended status: Standards Track
Expires: May 3, 2018

A. Gonzalez Prieto
VMware
E. Voit
Cisco Systems
A. Clemm
Huawei
E. Nilsen-Nygaard
A. Tripathy
Cisco Systems
October 30, 2017

NETCONF Support for Event Notifications
draft-ietf-netconf-netconf-event-notifications-06

Abstract

This document provides a NETCONF binding for
[I-D.[draft-ietf-netconf-subscribed-notifications](#)]. Included are:

- o Transport mappings for subscription RPCs, state change notifications, and notification messages
- o Functionality which must be supported with NETCONF
- o Examples in appendices

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 3, 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	3
3.	Interleave Capability	3
4.	Mandatory stream and datastore support	3
5.	Transport connectivity	4
5.1.	Dynamic Subscriptions	4
5.2.	Configured Subscriptions	4
6.	Notification Messages	4
7.	Security Considerations	4
8.	Acknowledgments	5
9.	References	5
9.1.	Normative References	5
9.2.	Informative References	6
Appendix A.	Examples	6
A.1.	Event Stream Discovery	6
A.2.	Dynamic Subscriptions	7
A.3.	Configured Subscriptions	12
A.4.	Subscription State Notifications	17
Appendix B.	Changes between revisions	19
B.1.	v05 to v06	19
B.2.	v03 to v04	19
B.3.	v01 to v03	20
B.4.	v00 to v01	20
	Authors' Addresses	20

[1.](#) Introduction

This document defines a binding for notification message delivery for [I-D.[draft-ietf-netconf-subscribed-notifications](#)] transported over the NETCONF protocol [[RFC6241](#)]. In addition, as [I-D.[ietf-netconf-yang-push](#)] is itself built upon

[I-D.[draft-ietf-netconf-subscribed-notifications](#)], this document enables a NETCONF client to maintain a subset/extract of an actively changing YANG datastore located on a NETCONF server.

This document is broken into two main parts. The first contains normative requirements which are incremental to [I-D.[draft-ietf-netconf-subscribed-notifications](#)] when NETCONF transport is used. The second are examples and are included as appendices.

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 [I-D.[draft-ietf-netconf-subscribed-notifications](#)]: notification message, stream, publisher, receiver, subscriber, subscription, configured subscription.

3. Interleave Capability

To support multiple subscriptions on a single session, a NETCONF publisher MUST support the :interleave capability as defined in [[RFC5277](#)]. Such support MUST be indicated by the following capability: "urn:ietf:params:netconf:capability:interleave:1.0". Advertisement of this capability along with support [I-D.[draft-ietf-netconf-subscribed-notifications](#)] will indicate that a NETCONF publisher is able to receive, process, and respond to NETCONF requests and [I-D.[draft-ietf-netconf-subscribed-notifications](#)] subscription operations on a session with active subscriptions.

4. Mandatory stream and datastore support

A NETCONF publisher supporting [I-D.[draft-ietf-netconf-subscribed-notifications](#)] MUST support the "NETCONF" event stream identified in that draft.

A NETCONF publisher supporting [I-D.[ietf-netconf-yang-push](#)] MUST support the "running" datastore as defined by [I-D.[draft-ietf-netmod-revised-datastores](#)].

5. Transport connectivity

5.1. Dynamic Subscriptions

For dynamic subscriptions, if the NETCONF session involved with the "establish-subscription" terminates, the subscription MUST be deleted.

5.2. Configured Subscriptions

For a configured subscription, there is no guarantee a transport session is currently in place with associated receiver(s). So where a configured subscription has a receiver in the connecting state, but no NETCONF transport exists to that receiver, the publisher MUST be able to initiate a NETCONF transport session via NETCONF call home [\[RFC8071\]](#), [section 4.1](#) to that receiver. Until NETCONF connectivity is established and a subscription-started state change notification is successfully sent, that receiver MUST remain in its status of a "connecting".

If the call home fails because the publisher receives receiver credentials which are subsequently declined as part [\[RFC8071\]](#), [Section 4.1](#), step S5 authentication, then that receiver MUST be assigned a "suspended" status.

If the call home fails to establish for any other reason, the publisher MAY leave the receiver in a "connecting" status, and retry the call home at a future time. Alternatively, the publisher MAY place the receiver into a "suspended" status after a predetermined number of call home attempts.

NETCONF Transport session connectivity SHOULD be verified via [Section 4.1](#), step S7.

Failure of an active NETCONF session MUST reset the restart the call home process, and return the receiver to "connecting".

6. Notification Messages

Notification messages transported over NETCONF will be identical in format and content to those encoded using one-way operations defined within [\[RFC5277\]](#), [section 4](#).

7. Security Considerations

Notification messages (including state change notifications) are never sent before the NETCONF capabilities exchange has completed.

If a malicious or buggy 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 suspect underlying NETCONF sessions. The publisher MAY also suspend or terminate a subset of the active subscriptions on the NETCONF session.

The NETCONF Authorization Control Model [[RFC6536](#)] SHOULD be used to control and restrict authorization of subscription configuration.

8. 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.

9. References

9.1. 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-06](#) (work in progress), October 2017.

[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", October 2017, <<https://datatracker.ietf.org/doc/draft-ietf-netconf-yang-push/>>.

[I.D.[draft-ietf-netmod-revised-datastores](#)]

Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture", [draft-ietf-netmod-revised-datastores-04](#) (work in progress), August 2017.

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

[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>>.
- [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>>.

9.2. Informative References

- [I-D.[draft-ietf-netconf-notification-messages](#)]
Voit, Eric., Clemm, Alexander., Bierman, A., and T. Jenkins, "YANG Notification Headers and Bundles", September 2017, <<https://datatracker.ietf.org/doc/draft-ietf-netconf-notification-messages>>.

Appendix A. Examples

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](#)]. Any reply will include the stream identities supported on the NETCONF publisher which may be available to that client.

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-subscribed-notifications"/>
      </filter>
    </get>
  </rpc>
```

Figure 1: Get streams request

After such a request, the NETCONF publisher returns a list of event streams available. In the example reply below, the list contains just the NETCONF stream.

```
<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-subscribed-notifications">
      <name>NETCONF</name>
    </streams>
  </data>
</rpc-reply>
```

Figure 2: Get streams response

[A.2.](#) Dynamic Subscriptions

The dynamic subscription RPCs and interactions operation are defined in [I-D.[draft-ietf-netconf-subscribed-notifications](#)] and enhanced in [I-D.[ietf-netconf-yang-push](#)].

[A.2.1.](#) Establishing Dynamic Subscriptions

An example of establish-subscription interactions over NETCONF transport for a sample subscription is described below:


```
<rpc netconf: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>
      <name>NETCONF</name>
      <xpath-filter xmlns:ex="http://example.com/events">
        /ex:foo
      </xpath-filter>
    </stream>
  </establish-subscription>
</rpc>
```

Figure 3: establish-subscription over NETCONF

If the NETCONF publisher can satisfy the request, the publisher 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-subscribed-notifications">
    ok
  </subscription-result>
  <identifier
    xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications">
    22
  </identifier>
</rpc-reply>
```

Figure 4: Successful establish-subscription

If the NETCONF publisher cannot satisfy the request, or subscriber has no authorization to establish the subscription, the publisher will send a negative "subscription-result" element. For instance:

```
<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-subscribed-notifications">
    stream-unavailable
  </subscription-result>
</rpc-reply>
```

Figure 5: Unsuccessful establish subscription

To get an idea of the interaction model, the following figure shows two separate establish subscriptions RPC being made. The first is given subscription id 22, the second, id 23.

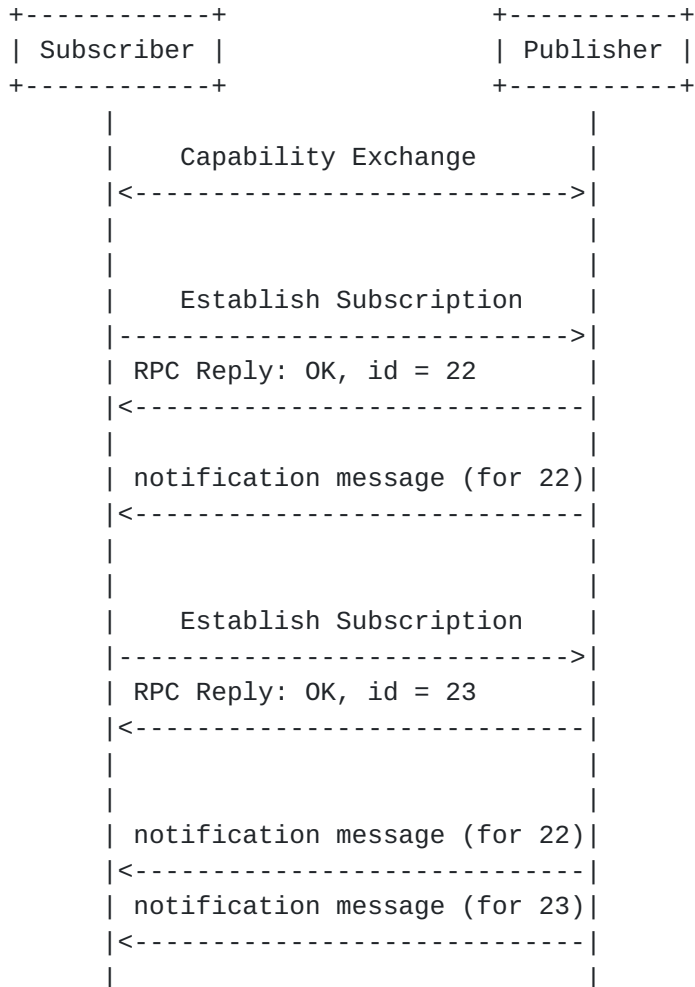


Figure 6: Multiple subscription establishments over a single NETCONF session

In the example above, it is important to note that the subscription ids of 22 and 23 are not included in the notification messages of [I-D.[draft-ietf-netconf-subscribed-notifications](#)]. However because [I-D.[ietf-netconf-yang-push](#)] has defined its own notifications, subscription identifiers are available within those notification messages. With the availability of [I-D.[draft-ietf-netconf-notification-messages](#)], all notification messages will be able to transport a subscription identifier.

[A.2.2.](#) Modifying Dynamic Subscriptions

The following demonstrates modifying a dynamic subscription. Consider a subscription from [[I-D.ietf-netconf-yang-push](#)]. An established may have a new filter applied. The desired modification is the application of a new filter.

```
<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">
    <yp:datastore>
      <yp:xpath-filter xmlns="http://example.com/datastore">
        /interfaces-state/interface/oper-status
      </yp:xpath-filter>
    </yp:datastore>
    <identifier>
      22
    </identifier>
  </modify-subscription>
</rpc>
```

Figure 7: Subscription modification

If the NETCONF publisher can satisfy the request, the publisher sends a positive "subscription-result". 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-subscribed-notifications">
    ok
  </subscription-result>
</rpc-reply>
```

Figure 8: Successful modify-subscription

If the NETCONF publisher cannot satisfy the request, the publisher sends a negative "subscription-result" element. Its contents and semantics match those from an establish-subscription request.

To get an idea of the interaction model, the following figure shows a successful RPC modification request to subscription with an identifier of 22.

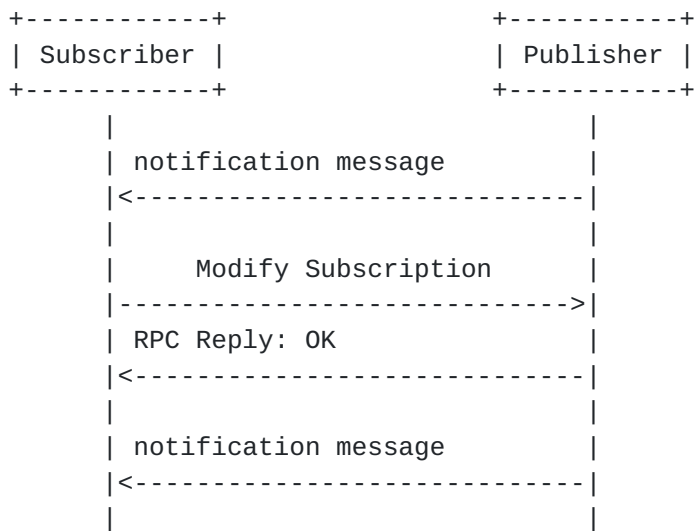


Figure 9: Interaction model for successful subscription modification

A.2.3. Deleting Dynamic Subscriptions

The following demonstrates deleting a subscription.

```

<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">
    <identifier>22</identifier>
  </delete-subscription>
</rpc>

```

Figure 10: Delete subscription

If the NETCONF publisher can satisfy the request, the publisher sends an OK element. For example:

```

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

```

Figure 11: Successful delete subscription

If the NETCONF publisher cannot satisfy the request, the publisher sends an error-rpc element indicating the modification didn't work. One way this could happen is if an existing valid subscription identifier was given, but that subscription 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-path
      xmlns:sn="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications">
      sn:identifier
    </error-path>
    <error-message xml:lang="en">
      no-such-subscription
    </error-message>
  </rpc-error>
</rpc-reply>
```

Figure 12: Unsuccessful delete subscription

[A.3.](#) Configured Subscriptions

Configured subscriptions may be established, modified, and deleted using configuration operations against the top-level subtree of [[I-D.draft-ietf-netconf-subscribed-notifications](#)] or [[I-D.ietf-netconf-yang-push](#)].

In this section, we present examples of how to manage the configuration subscriptions using a NETCONF client. Key differences from dynamic subscriptions over NETCONF is that subscription lifetimes are decoupled from NETCONF sessions.

[A.3.1.](#) Creating Configured Subscriptions

For subscription creation, a NETCONF client may send:


```
<rpc message-id="201"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
  xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <subscription-config
      xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications">
      <subscription>
        <identifier>22</identifier>
        <encoding>encode-xml</encoding>
        <stream>
          <name>NETCONF</name>
          <receiver>
            <address>1.2.3.4</address>
            <port>1234</port>
          </receiver>
        </stream>
      </subscription>
    </subscription-config>
  </edit-config>
</rpc>
```

Figure 13: Create a configured subscription

If the request is accepted, the publisher would reply:

```
<rpc-reply message-id="201"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

Figure 14: Response to a successful configuration subscription establishment

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


```

<rpc-reply message-id="201"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <rpc-error>
    <error-type>application</error-type>
    <error-tag>resource-denied</error-tag>
    <error-severity>error</error-severity>
    <error-message xml:lang="en">
      Temporarily the publisher cannot serve this
      subscription due to the current workload.
    </error-message>
  </rpc-error>
</rpc-reply>

```

Figure 15: Response to a failed configured subscription establishment

After a subscription has been created, NETCONF connectivity to each receiver's IP address and port will be established if it does not already exist. This will be accomplished via [\[RFC8071\]](#).

To get an idea of the interaction model, the following figure shows a successful configuration based creation of a subscription.

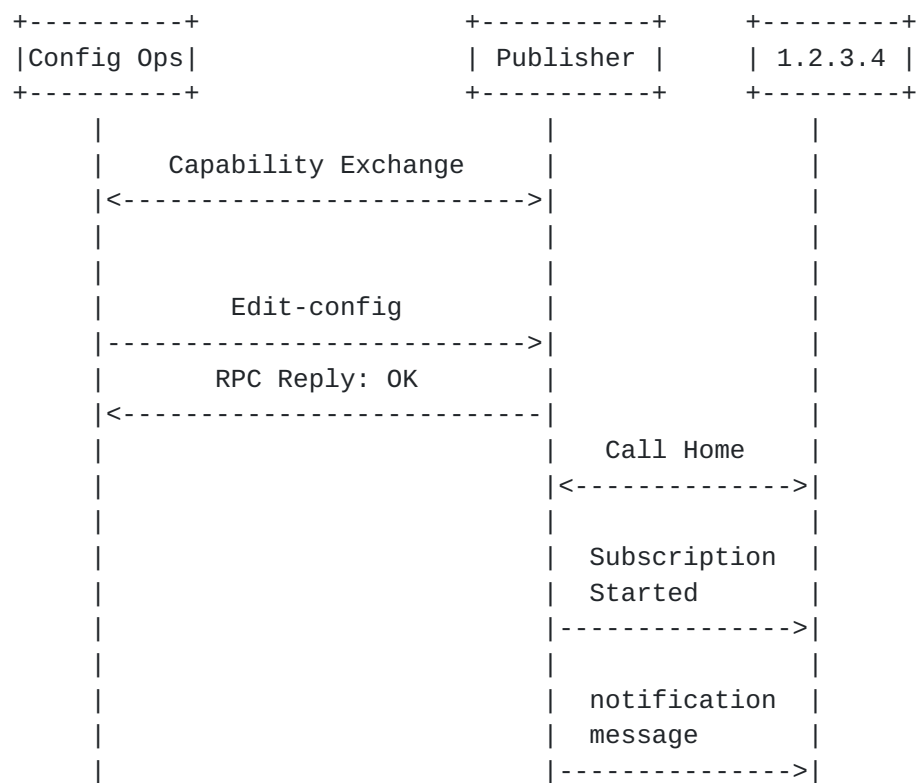


Figure 16: Interaction model for configured subscription establishment

[A.3.2.](#) Modifying Configured Subscriptions

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, here adding a second receiver:

```
<rpc message-id="202"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
  xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <subscription-config
      xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications">
      <subscription>
        <identifier>
          1922
        </identifier>
        <receiver>
          <address>
            1.2.3.5
          </address>
          <port>
            1234
          </port>
        </receiver>
      </subscription>
    </subscription-config>
  </edit-config>
</rpc>
```

Figure 17: Modify configured subscription

If the request is accepted, the publisher would reply:

```
<rpc-reply message-id="202"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

Figure 18: A successful configured subscription modification

And the previous interaction model would be extended as follows.



Figure 19: Interaction model for configured subscription modification

Note in the above that in the specific example above, modifying a configured subscription actually resulted in subscription-started notification. If the edit of the configuration had also added a filter, a separate modify-subscription would have gone to the original receiver.

[A.3.3. Deleting Configured Subscriptions](#)

Configured subscriptions can be deleted using configuration operations against the top-level subtree subscription-config. Deleting the subscription above would result in the following flow impacting all receivers.



Figure 20: Interaction model for configured subscription deletion

A.4. Subscription State Notifications

A publisher will send subscription state notifications according to the definitions within [I-D.[draft-ietf-netconf-subscribed-notifications](#)]).

A.4.1. subscription-started and subscription-modified

A subscription-started over NETCONF 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-started
    xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications"/>
    <identifier>39</identifier>
    <encoding>encode-xml</encoding>
    <stream>
      <name>NETCONF</name>
      <xpath-filter xmlns:ex="http://example.com/events">
        /ex:foo
      </xpath-filter>
    </stream>
  </subscription-started/>
</notification>
```

Figure 21: subscription-started subscription state notification

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

A.4.2. subscription-completed, subscription-resumed, and replay-complete

A subscription-completed would look like:

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

Figure 22: subscription-completed notification in XML

The subscription-resumed and replay-complete are virtually identical, with "subscription-completed" simply being replaced by "subscription-resumed" and "replay-complete" in both encodings.

A.4.3. 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">
    <identifier>39</identifier>
    <error-id>
      unsupportable-volume
    </error-id>
  </subscription-terminated>
</notification>
```

Figure 23: subscription-terminated subscription state notification

The subscription-suspended is virtually identical, with "subscription-terminated" simply being replaced by "subscription-suspended".

Appendix B. Changes between revisions

(To be removed by RFC editor prior to publication)

B.1. 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.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

Eric Voit
Cisco Systems

Email: evoit@cisco.com

Alexander Clemm
Huawei

Email: ludwig@clemm.org

Einar Nilsen-Nygaard
Cisco Systems

Email: einarnn@cisco.com

Ambika Prasad Tripathy
Cisco Systems

Email: ambtripa@cisco.com

