NETCONF
Internet-Draft
Intended status: Standards Track
Expires: December 17, 2016

A. Gonzalez Prieto
A. Clemm
E. Voit
E. Nilsen-Nygaard
A. Tripathy
Cisco Systems
S. Chisholm
Ciena
H. Trevino
Cisco Systems
June 15, 2016

NETCONF Support for Event Notifications
draft-gonzalez-netconf-event-notifications-00

Abstract

This document defines the support of [event-notifications] by the Network Configuration protocol (NETCONF). [event-notifications] describes capabilities and operations for providing asynchronous message notification delivery. This document discussed how to provide them on top of NETCONF. The capabilities and operations defined between this document and [event-notifications] are intended to obsolete RFC 5277.

Table of Contents

1. Introduction

[RFC6241] can be conceptually partitioned into four layers:

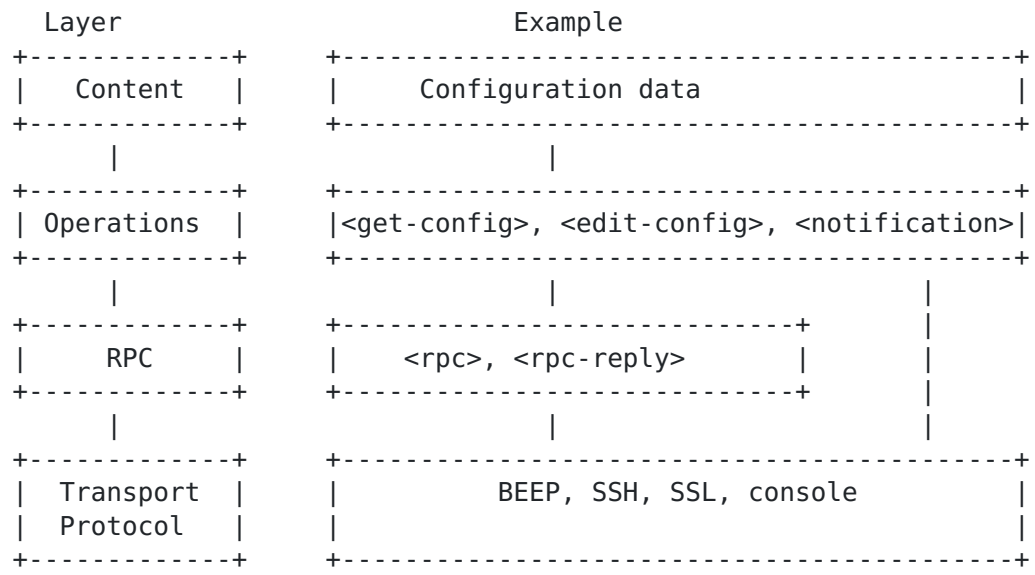


Figure 1: NETCONF layer architecture

This document defines mechanisms that provide an asynchronous message notification delivery service for the NETCONF protocol [RFC6241] based on [event-notifications]. This is an optional capability built on top of the base NETCONF definition.

[event-notifications] and this document enhance the capabilities of RFC 5277 while maintaining backwards capability with existing implementations. It is intended that a final version of this document might obsolete [RFC5277].

The enhancements over [RFC5277] include the ability to terminate subscriptions without terminating the client session, to modify existing subscriptions, and to have multiple subscriptions on a NETCONF session.

These enhancements do not affect [RFC5277] clients that do not support these particular subscription requirements.

[event-notifications] covers the following functionality:

- Ability to subscribe to event notifications using two mechanisms: dynamic and configuration subscriptions.

- Ability to subscribe to event notifications using two mechanisms: dynamic and configuration subscriptions.

- Ability to negotiate acceptable subscription parameters.

- Ability to filter the subset of notifications to be pushed with stream-specific semantics.
- Ability to support multiple encodings for the notification.
- Mechanism to communicate the notifications.
- Ability to replay locally logged notifications.

To support this functionality, NETCONF agents must implement the operations, configuration and operational state defined in [event-notifications]. In addition, they need to:

- support multiple subscriptions over a single NETCONF session.
- support a revised definition of the default NETCONF stream
- be backwards compatible with RFC 5277 implementations.

1.1. 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].

1.1.1. Event Notifications

The following terms are defined in [event-notifications]:

- Event
- Event notification
- Stream (also referred to as "event stream")
- Subscriber
- Receiver
- Subscription
- Filter
- Dynamic subscription
- Configured subscription

1.1.2. NETCONF

The following terms are defined in [RFC6241] :

- Operation
- RPC: remote procedure call

1.1.3. NETCONF Access Control

The following terms are defined in [RFC6536] :

- NACM: NETCONF Access Control Model

1.2. Solution Overview

[event-notifications] defines mechanisms that provide an asynchronous message notification delivery service. This document discusses its realization on top of the NETCONF protocol [RFC6241].

The functionality to support is defined in [event-notifications]. It is formalized in a set of yang models. The mapping of yang constructs into NETCONF is described in [RFC6020].

Supporting [event-notifications] requires enhancements and modifications in NETCONF. The key enhacement is suporting multiple subscriptions on a NETCONF session. A key modification is the definition of the NETCONF stream

These enhancements do not affect [RFC5277] clients that do not support [event-notifications].

2. Solution

In this section, we describe and exemplify how [event-notifications] must be supported over NETCONF.

2.1. Event Streams

In the context of NETCONF, an event stream is a set of events available for subscription from a NETCONF server. It is out of the scope of this document to identify a) how streams are defined, b) how events are defined/generated, and c) how events are assigned to streams.

The following is a high-level description of the flow of a notification. Note that it does not mandate and/or preclude an implementation. As events are raised, they are assigned to streams.

An event may be assigned to multiple streams. The event is distributed to subscribers and receivers based on the current subscriptions and access control. Access control is needed because if any receiver of that subscription does not have permission to receive an event, then it never makes it into a notification, and processing of the event is completed for that subscription.

2.2. Event Stream Discovery

A NETCONF client can retrieve the list of available event streams from a NETCONF server using the <get> operation. The reply contains the elements defined in the YANG model under the container "/streams", which includes the stream identifier.

The following example ilustrates 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-event-notifications"/>
      </filter>
  </get>
</rpc>
```

Figure 2: Get streams

The NETCONF server returns a list of event streams available for subscription. In this example, the list contains the NETCONF, SNMP, and syslog-critical 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-event-notifications">
      <stream>NETCONF</stream>
      <stream>SNMP</stream>
      <stream>syslog-critical</stream>
      <stream>NETCONF</stream>
      </streams>
  </data>
</rpc-reply>
```

Figure 3: Get streams response

2.2.1. Backwards Compatibility

In order to maintain backwards compatibility, clients that only support [RFC5277] can retrieve the list of available event streams executing a <get> operation against the container "/netconf/streams".

The following example ilustrates this mechanism.

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

Figure 4: Get streams (backwards compatibility)

The NETCONF server returns a list of event streams available for subscription. In this example, the list contains the NETCONF, SNMP, and syslog-critical streams.

```
<rpc-reply message-id="101"
        xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <netconf
            xmlns="urn:ietf:params:xml:ns:netmod:notification">
      <streams>
        <stream>
          <name>
            NETCONF
          </name>
          <description>
            default NETCONF event stream
          </description>
          <replaySupport>
            true
          </replaySupport>
          <replayLogCreationTime>
            2016-02-05T00:00:00Z
          </replayLogCreationTime>
        </stream>
        <stream>
          <name>
```

```
              SNMP
            </name>
            <description>
              SNMP notifications
            </description>
            <replaySupport>
              false
            </replaySupport>
          </stream>
          <stream>
            <name>
              syslog-critical
            </name>
            <description>
              Critical and higher severity
            </description>
            <replaySupport>
              true
            </replaySupport>
            <replayLogCreationTime>
              2007-07-01T00:00:00Z
            </replayLogCreationTime>
          </stream>
        </streams>
      </netconf>
    </data>
  </rpc-reply>
```

Figure 5: Get streams response (backwards compatibility)

2.3. Default Event Stream

A NETCONF server implementation supporting the notification capability MUST support the "NETCONF" notification event stream. This stream contains all NETCONF XML event notifications supported by the NETCONF server, except for those belonging only to streams that explicitly indicate that they must be excluded from the NETCONF stream. The exact string "NETCONF" is used during the advertisement of stream support during the <get> operation on <streams> and during the <create-subscription> and <establish-subscription> operations.

2.4. Creating a Subscription

The following illustrates the creation of a simple subscription.

2.4.1. Usage Example

The following demonstrates dynamically creating a subscription. This operation is fully defined in [RFC5277]

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

Figure 6: Create subscription

2.4.2. Positive Response

If the NETCONF server can satisfy the request, the server sends an <ok> element.

```
<netconf:rpc netconf:message-id="102"
       xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
  <create-subscription
          xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
    <filter netconf:type="xpath"
               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')]"/>
  </create-subscription>
</netconf:rpc>

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

Figure 7: Successful create subscription

2.4.3. Negative Response

If the request cannot be completed for any reason, an <rpc-error> element is included within the <rpc-reply>. Subscription requests can fail for several reasons including if a filter with invalid syntax is provided or if the name of a non-existent stream is provided.

If a stopTime is specified in a request without having specified a startTime, the following error is returned:

```
Tag: missing-element
Error-type: protocol
Severity: error
Error-info: <bad-element>: startTime
Description: An expected element is missing.
```

Figure 8: Create subscription missing an element

If the optional replay feature is requested but the NETCONF server does not support it, the following error is returned:

```
Tag: operation-failed
Error-type: protocol
Severity: error
Error-info: none
Description: Request could not be completed because the
             requested operation failed for some reason
             not covered by any other error condition.
```

Figure 9: Create subscription pperation failed

If a stopTime is requested that is earlier than the specified startTime, the following error is returned:

```
Tag: bad-element
Error-type: protocol
Severity: error
Error-info: <bad-element>: stopTime
Description: An element value is not correct;
             e.g., wrong type, out of range, pattern mismatch.
```

Figure 10: Create subscription incorrect stopTime

If a startTime is requested that is later than the current time, the following error is returned:

```
Tag: bad-element
Error-type: protocol
Severity: error
Error-info: <bad-element>: startTime
Description: An element value is not correct;
             e.g., wrong type, out of range, pattern mismatch.
```

Figure 11: Create subscription incorrect startTime

2.5. Establishing a Subscription

2.5.1. Usage Example

The following illustrates the establishment of a simple subscription.

```
<netconf:rpc message-id="101"
       xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
  <establish-subscription
         xmlns="urn:ietf:params:xml:ns:netconf:notification:1.1">
  </establish-subscription>
</netconf:rpc>
```

Figure 12: Establish subscription

2.5.2. 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.

```
<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:netconf:notification:1.1">
    <filter netconf:type="xpath"
               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>

<rpc-reply message-id="102"
       xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <subscription-result
         xmlns="urn:ietf:params:xml:ns:netconf:notification:1.1">
    ok
  </subscription-result>
  <subscription-id
         xmlns="urn:ietf:params:xml:ns:netconf:notification:1.1">
    52
  </subscription-id>
</rpc-reply>
```

Figure 13: Successful establish subscription

2.5.3. Negative Response

If the NETCONF server cannot satisfy the request, the server sends a negative <subscription-result> element.

If the client has no authorization to establish the subscription, the <subscription-result> indicates an authorization error. For instance:

```
<netconf:rpc netconf:message-id="103"
       xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
  <establish-subscription
          xmlns="urn:ietf:params:xml:ns:netconf:notification:1.1">
    <stream>foo</stream>
  </establish-subscription>
</netconf:rpc>

<rpc-reply message-id="103"
       xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <subscription-result
          xmlns="urn:ietf:params:xml:ns:netconf:notification:1.1">
    error-data-not-authorized
  </subscription-result>
</rpc-reply>
```

Figure 14: Unsuccessful establish subscription

If the request is rejected because the server is not able to serve it, the server SHOULD include in the returned error what subscription parameters would have been accepted for the request when it was processed. However, they are no guarantee that subsequent requests with those parameters for this client or others will be accepted. For instance, consider a subscription from [netconf-yang-push], which augments the establish-subscription with some additional parameters, including "period". If the client requests a period the NETCONF server cannot serve, the 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:netconf:notification:1.1">
    <stream>push-update</stream>
    <filter netconf:type="xpath"
            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">
      500
    </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:netconf:notification:1.1">
    error-insufficient-resources
  </subscription-result>
  <period
        xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
    2000
  </period>
</rpc-reply>
```

Figure 15: Subscription establishment negotiation

Subscription requests will fail if a filter with invalid syntax is provided or if the name of a non-existent stream is provided.

2.5.4. Multiple Subscriptions over a Single NETCONF Session

Note that [event-notifications] requires supporting multiple subscriptions over a single NETCONF session. In contrast, [RFC5277] mandated server to return an error when a create-subscription was sent while a subscription was active on that session.

2.5.5. Message Flow Examples

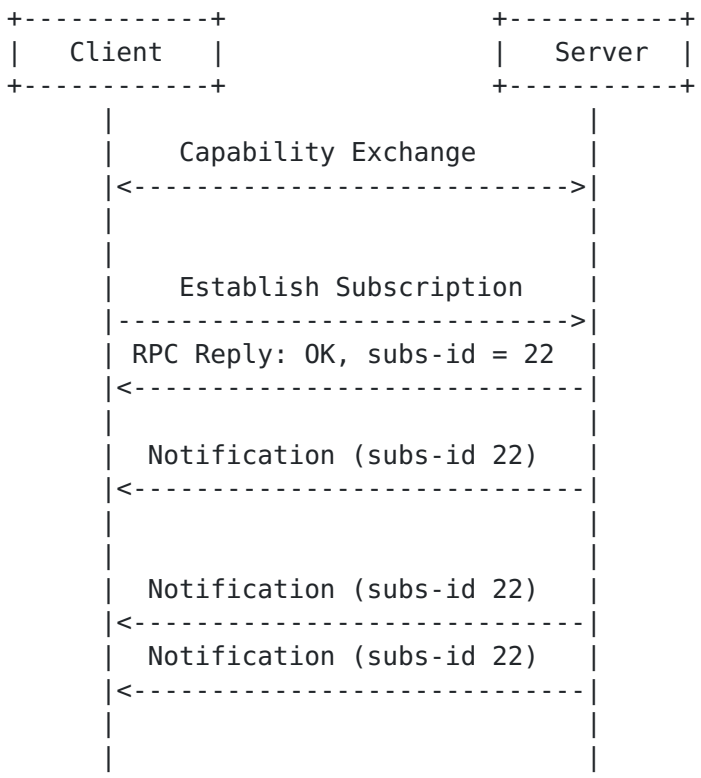


Figure 16: Message flow for subscription establishment

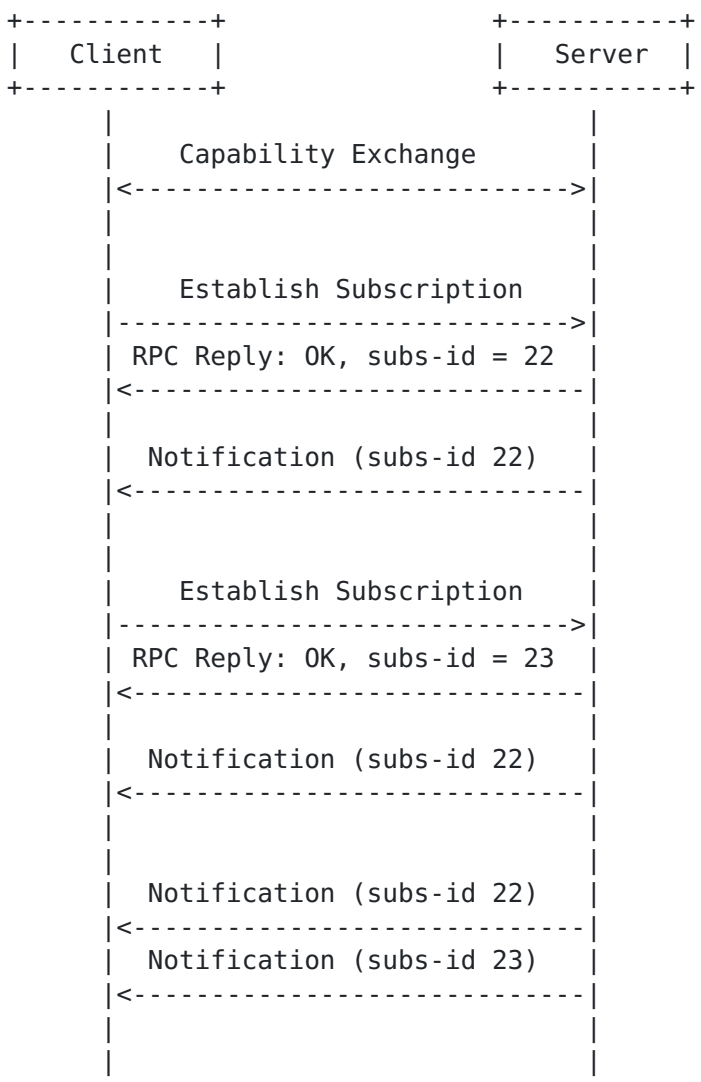


Figure 17: Message Flow for multiple subscription establishments over a single session

2.6. Modifying a Subscription

2.6.1. Usage Example

The following demonstrates modifying a subscription. Consider a subscription from [netconf-yang-push], which augments the establish-subscription with some additional parameters, including "period". A subscription may be established 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:netconf:notification:1.1">
    <stream>push-update</stream>
    <filter netconf:type="xpath"
            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">
      500
    </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:netconf:notification:1.1">
    ok
  </subscription-result>
  <subscription-id
           xmlns="urn:ietf:params:xml:ns:netconf:notification:1.1">
    1922
  </subscription-id>
</rpc-reply>
```

Figure 18: Establish subscription to be modified

The subscription may be modified with:

```
<netconf:rpc message-id="102"
       xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
  <modify-subscription
          xmlns="urn:ietf:params:xml:ns:netconf:notification:1.1">
    <subscription-id>1922</subscription-id>
    <period>1000</period>
  </modify-subscription >
</netconf:rpc>

<rpc-reply message-id="102"
       xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <subscription-result
          xmlns="urn:ietf:params:xml:ns:netconf:notification:1.1">
    ok
  </subscription-result>
  <subscription-id
          xmlns="urn:ietf:params:xml:ns:netconf:notification:1.1">
    1922
  </subscription-id>
</rpc-reply>
```

Figure 19: Modify subscription

2.6.2. 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-id, which would be redundant.

```
<netconf:rpc message-id="102"
       xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
  <modify-subscription
          xmlns="urn:ietf:params:xml:ns:netconf:notification:1.1">
    <subscription-id>1922</subscription-id>
    <period
              xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
      1000
    </period>
  </modify-subscription >
</netconf:rpc>

<rpc-reply message-id="102"
       xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <subscription-result
          xmlns="urn:ietf:params:xml:ns:netconf:notification:1.1">
    ok
  </subscription-result>
</rpc-reply>
```

Figure 20: Successful modify subscription

2.6.3. 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 to an establish-subscription request. For instance:

```
<netconf:rpc message-id="102"
       xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
  <modify-subscription
          xmlns="urn:ietf:params:xml:ns:netconf:notification:1.1">
    <subscription-id>1922</subscription-id>
    <period xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
      100
    </period>
  </modify-subscription>
</netconf:rpc>

<rpc-reply message-id="102"
       xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <subscription-result
          xmlns="urn:ietf:params:xml:ns:netconf:notification:1.1">
    error-insufficient-resources
  </subscription-result>
  <period>500</period>
</rpc-reply>
```

Figure 21: Unsuccessful modify subscription

2.6.4. Message Flow Example

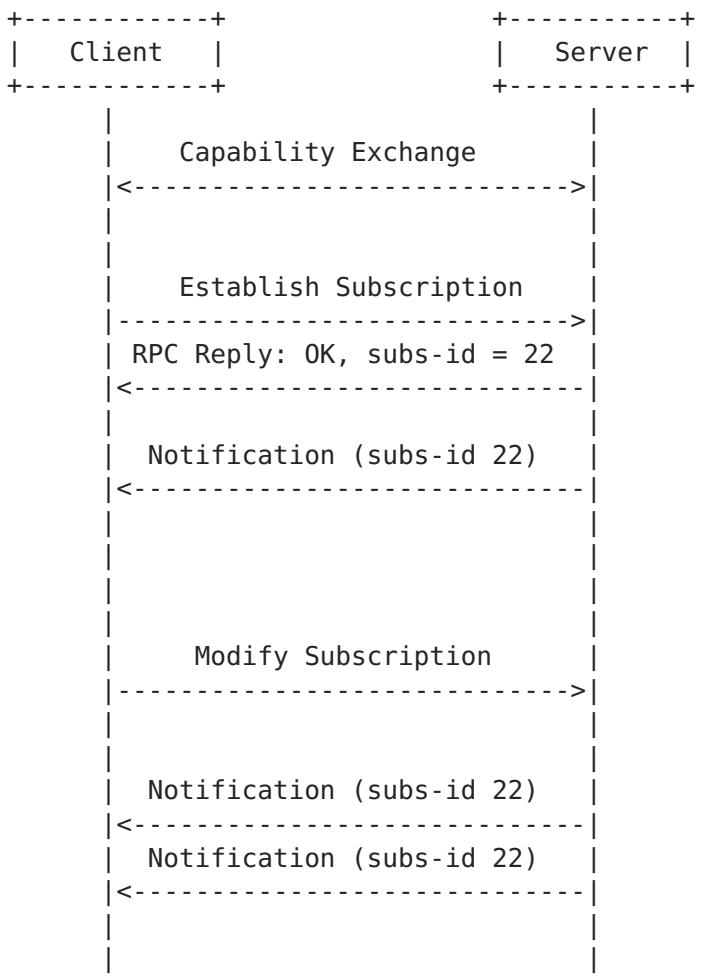


Figure 22: Message flow for subscription modification

2.7. Deleting a Subscription

2.7.1. 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:netconf:notification:1.1">
    <subscription-id>1922</subscription-id>
  </delete-subscription>
</netconf:rpc>
```

Figure 23: Delete subscription

2.7.2. Positive Response

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

```
<netconf:rpc message-id="103"
       xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
  <delete-subscription
          xmlns="urn:ietf:params:xml:ns:netconf:notification:1.1">
    <subscription-id>1922</subscription-id>
  </delete-subscription>
</netconf:rpc>

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

Figure 24: Successful delete subscription

2.7.3. Negative Response

If the NETCONF server cannot satisfy the request, the server sends an error-rpc element. For example:

```
<netconf:rpc message-id="103"
       xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
  <delete-subscription
          xmlns="urn:ietf:params:xml:ns:netconf:notification:1.1">
    <subscription-id>2017</subscription-id>
  </delete-subscription>
</netconf:rpc>

<rpc-reply message-id="101"
       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:t="urn:ietf:params:xml:ns:netconf:notification:1.1">
      /t:subscription-id
    </error-path>
    <error-message xml:lang="en">
      Subscription-id 2017 does not exist
    </error-message>
  </rpc-error>
</rpc-reply>
```

Figure 25: Unsuccessful delete subscription

2.7.4. Message Flow Example

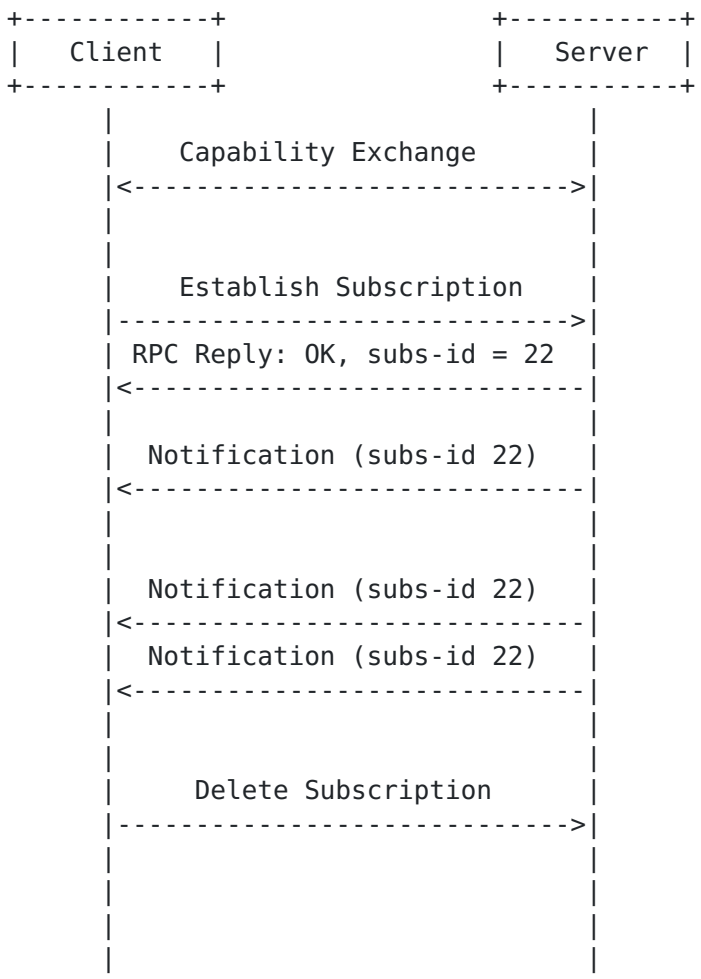


Figure 26: Message flow for subscription deletion

2.8. Configured Subscriptions

A configured subscription is a subscription installed via a configuration interface, and do not support negotiation.

Supporting configured subscriptions is optional and advertised during the capabilities exchange using the "configured-subscriptions" feature.

In this section, we present examples of how to manage configuration subscriptions using a NETCONF client.

2.8.1. Establishing a Configured Subscription

Subscriptions are established using configuration operations against the top-level subtree subscription-config.

For example at subscription establishment, a NETCONF client may send:

```
<rpc message-id="101"
       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:netconf:notification:1.1">
      <subscription>
        <subscription-id>
          1922
        </subscription-id>
        <stream>
          foo
        </stream>
        <receiver>
          <address>
            1.2.3.4
          </address>
          <port>
            1234
          </port>
        </receiver>
      </subscription>
    </subscription-config>
  </edit-config>
</rpc>
```

Figure 27: Establish static 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 28: Response to a successful static subscription establishment

```
   if the request is not accepted because the server cannot serve it,
   the server may reply:

   <rpc-reply 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 server cannot serve this
           subscription due to the current workload.
       </error-message>
     </rpc-error>
   </rpc-reply>
```

Figure 29: Response to a failed static subscription establishment

2.8.1.1. Message Flow Example

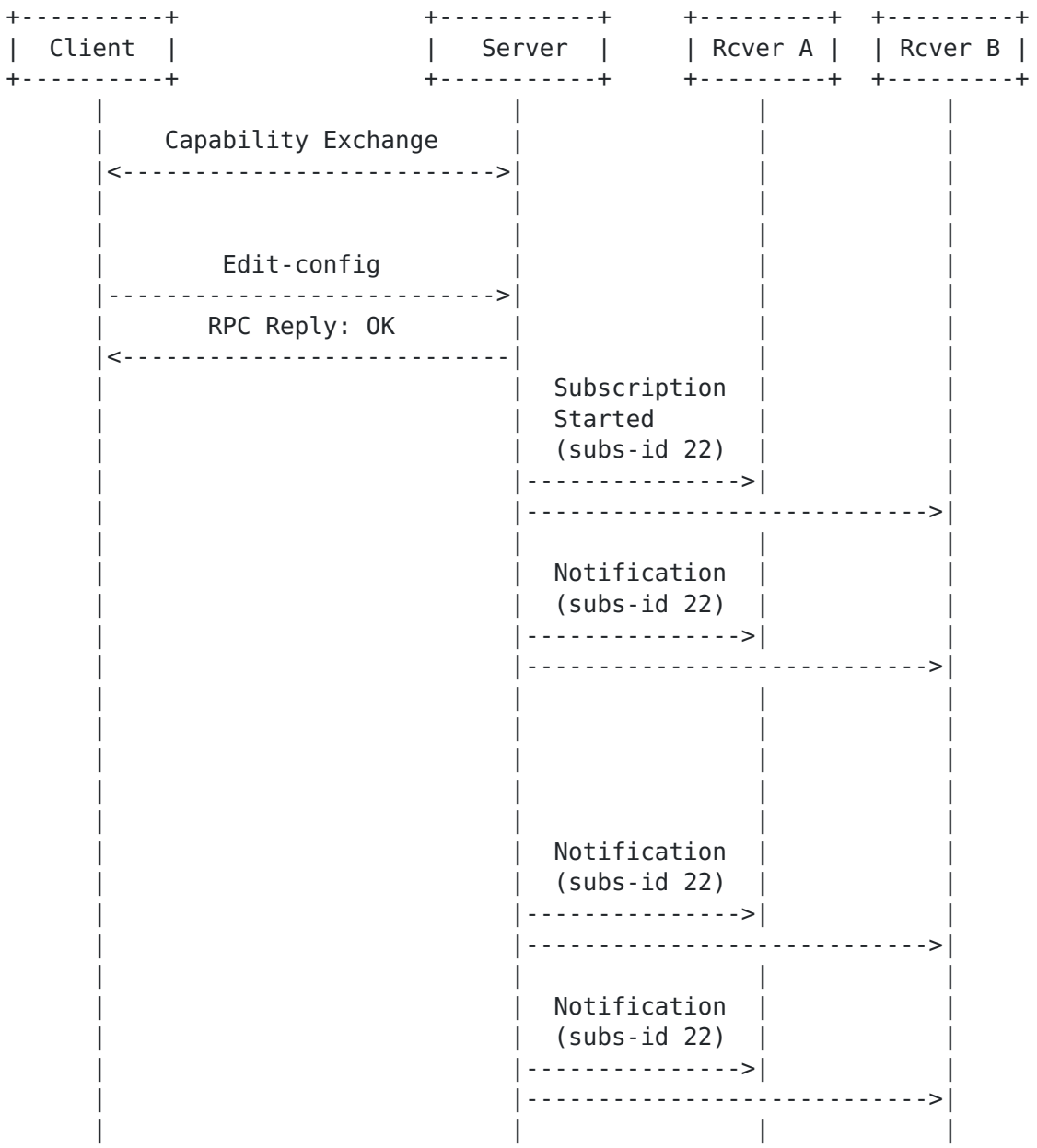


Figure 30: Message flow for subscription establishment (configured subscription)

2.8.2. 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"
       xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <subscription-config
            xmlns="urn:ietf:params:xml:ns:netconf:notification:1.1">
      <subscription>
        <subscription-id>
          1922
        </subscription-id>
        <stream>
          foo
        </stream>
        <receiver>
          <address>
            1.2.3.5
          </address>
          <port>
            1234
          </port>
        </receiver>
      </subscription>
    </subscription-config>
  </edit-config>
</rpc>
```

Figure 31: 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 32: Response to a successful configured subscription modification

2.8.2.1. Message Flow Example

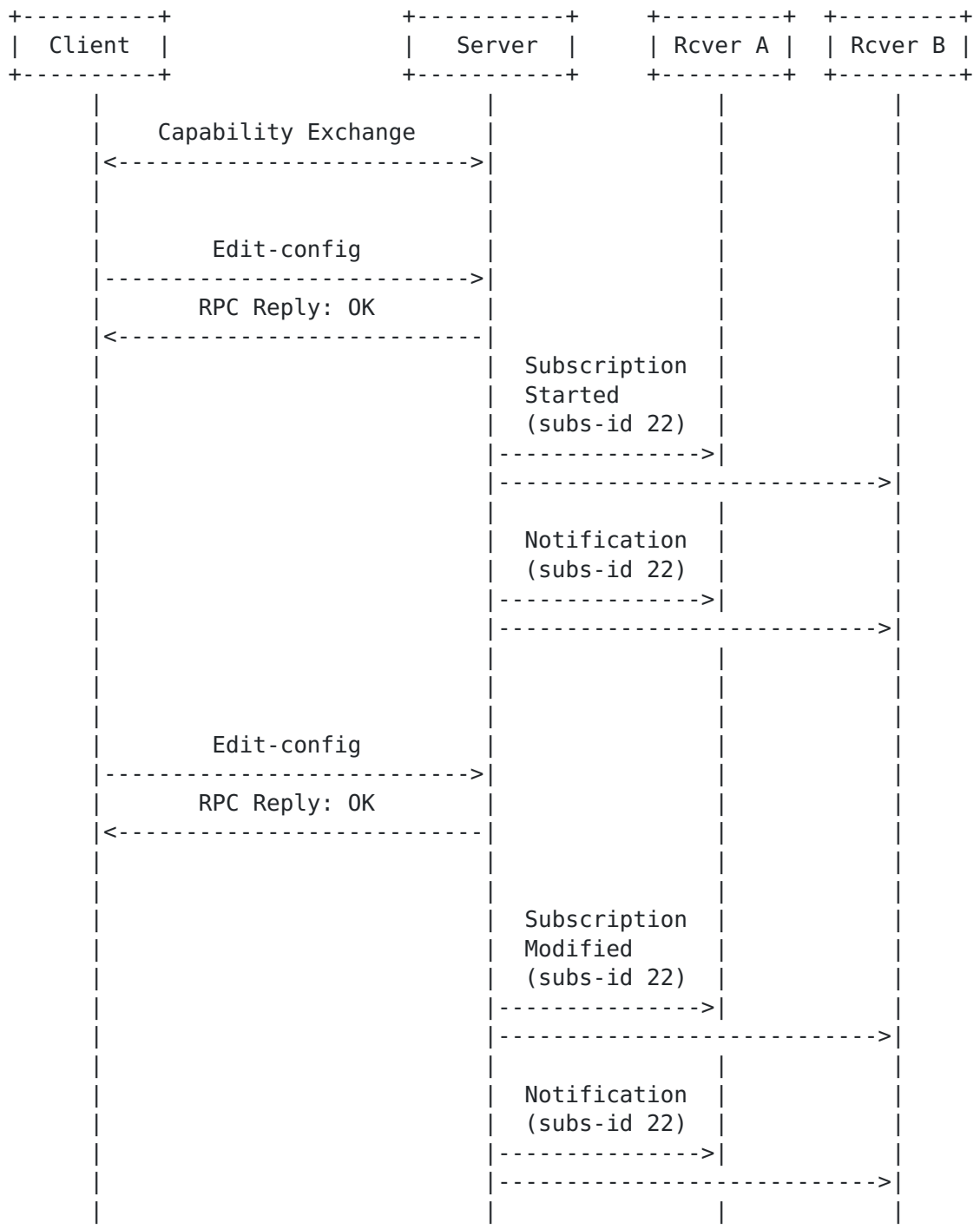


Figure 33: Message flow for subscription modification (configured subscription)

2.8.3. Deleting a Configured Subscription

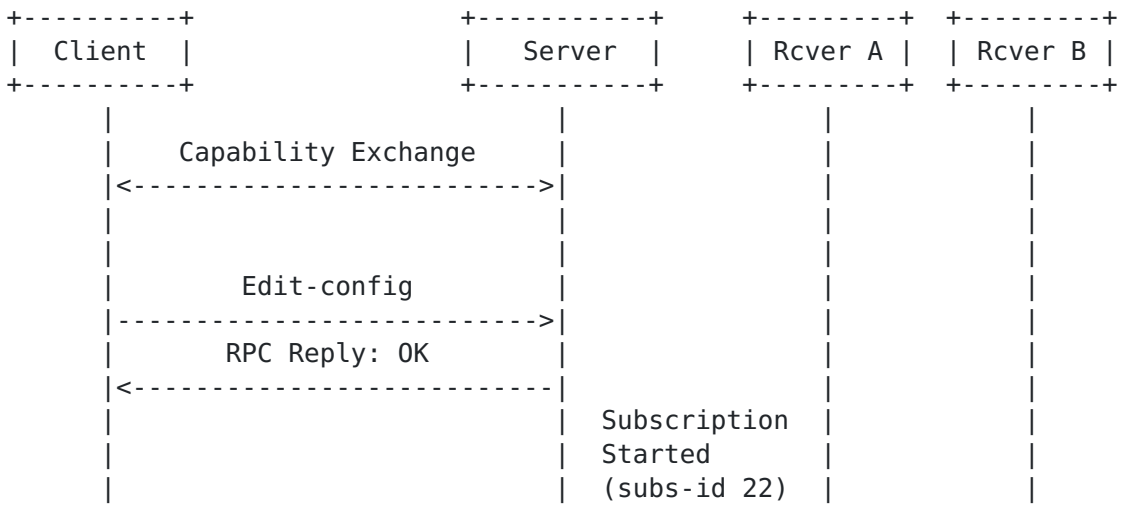
Subscriptions can be deleted using configuration operations against the top-level subtree subscription-config. For example:

```
<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:netconf:notification:1.1">
      <subscription xc:operation="delete">
        <subscription-id>
          1922
        </subscription-id>
      </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 34: Deleting a configured subscription

2.8.3.1. Message Flow Example



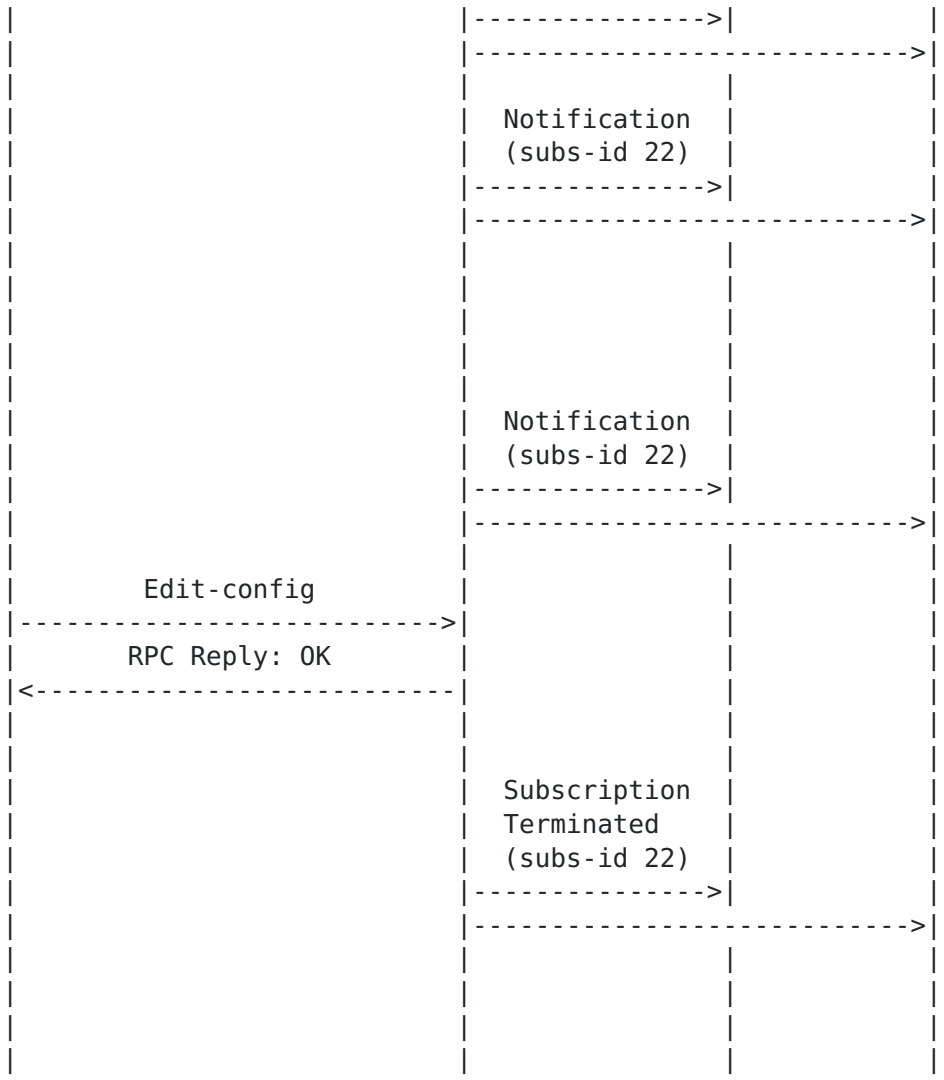


Figure 35: Message flow for subscription deletion (configured subscription)

2.9. Event (Data Plane) Notifications

Once a subscription has been set up, the NETCONF server sends (asynchronously) the event notifications from the subscribed stream. We refer to these as data plane notifications. For dynamic subscriptions set up via RPC operations, event notifications are sent over the NETCONF session used to create or establish the subscription. For static subscriptions, event notifications are sent over the specified connections.

An event notification is sent to the receiver(s) when an event of interest (i.e., meeting the specified filtering criteria) has occurred. An event notification is a complete and well-formed XML document. Note that <notification> is not a Remote Procedure Call (RPC) method but rather the top-level element identifying the one-way message as a notification. Note that event notifications never trigger responses.

The event notification always includes an <eventTime> element. It is the time the event was generated by the event source. This parameter is of type dateTime and compliant to [RFC3339]. Implementations must support time zones.

The event notification also contains notification-specific tagged content, if any. With the exception of <eventTime>, the content of the notification is beyond the scope of this document.

For the encodings other than XML, notifications include an additional XML element so that the notification is a well-formed XML. The element is <notification-contents-{encoding}>, E.g., <notification-contents-json>. That element contains the notification contents in the desired encoding

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 36: Definition of a data plane notification

```
<notification
       xmlns=" urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2007-09-01T10:00:00Z</eventTime>
  <link-failure xmlns="http://acme.example.com/system">
    <if-name>so-1/2/3.0</if-name>
    <if-admin-status>up</if-admin-status>
    <if-oper-status>down</if-oper-status>
  </link-failure>
</notification>
```

Figure 37: Data plane notification

The equivalent using JSON encoding would be

```
<notification
       xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2007-09-01T10:00:00Z</eventTime>
  <notification-contents-json>
       {
          "acme-system:link-failure": {
            "if-name": "so-1/2/3.0",
            "if-admin-status": "up",
            "if-oper-status": "down "
          }
        }
  </notification-contents-json>
</notification>
```

Figure 38: Data plane notification using JSON encoding

2.10. Control Plane Notifications

In addition to data plane notifications, a server may send control plane notifications (defined in [event-notifications]) to indicate to receivers that an event related to the subscription management has occurred. Control plane notifications cannot be filtered out. Next we exemplify them using both XML, and JSON encondings for the data:

2.10.1. replayComplete

```
<notification
       xmlns=" urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2007-09-01T10:00:00Z</eventTime>
  <replayComplete xmlns="urn:ietf:params:xml:ns:netmod:notification"/>
</notification>
```

Figure 39: replayComplete control plane notification

The equivalent using JSON encoding would be:

```
<notification
       xmlns=" urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2007-09-01T10:00:00Z</eventTime>
  <notification-contents-json>
       {
          "netmod-notif:replayComplete": { }
        }
  </notification-contents-json>
</notification>
```

Figure 40: replayComplete control plane notification (JSON encoding)

2.10.1.1. Message Flow Example

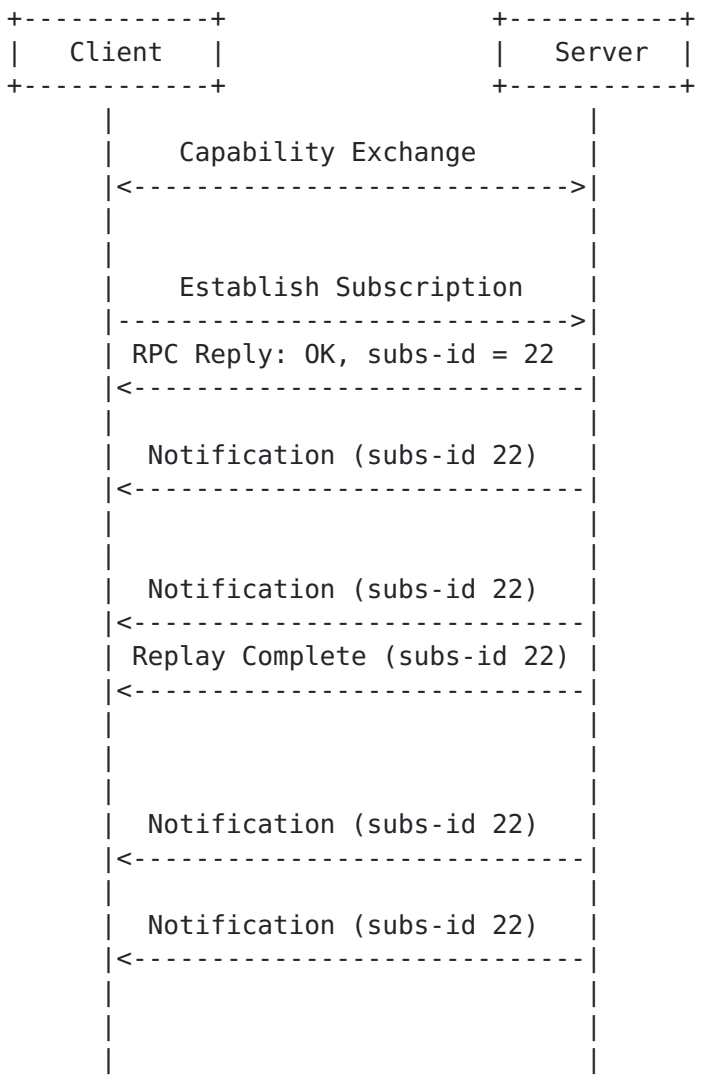


Figure 41: replayComplete notification

2.10.2. notificationComplete

```
<notification
    xmlns=" urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2007-09-01T10:00:00Z</eventTime>
  <notificationComplete
    xmlns="urn:ietf:params:xml:ns:netmod:notification"/>
</notification>
```

Figure 42: notificationComplete control plane notification

2.10.2.1. Message Flow Example

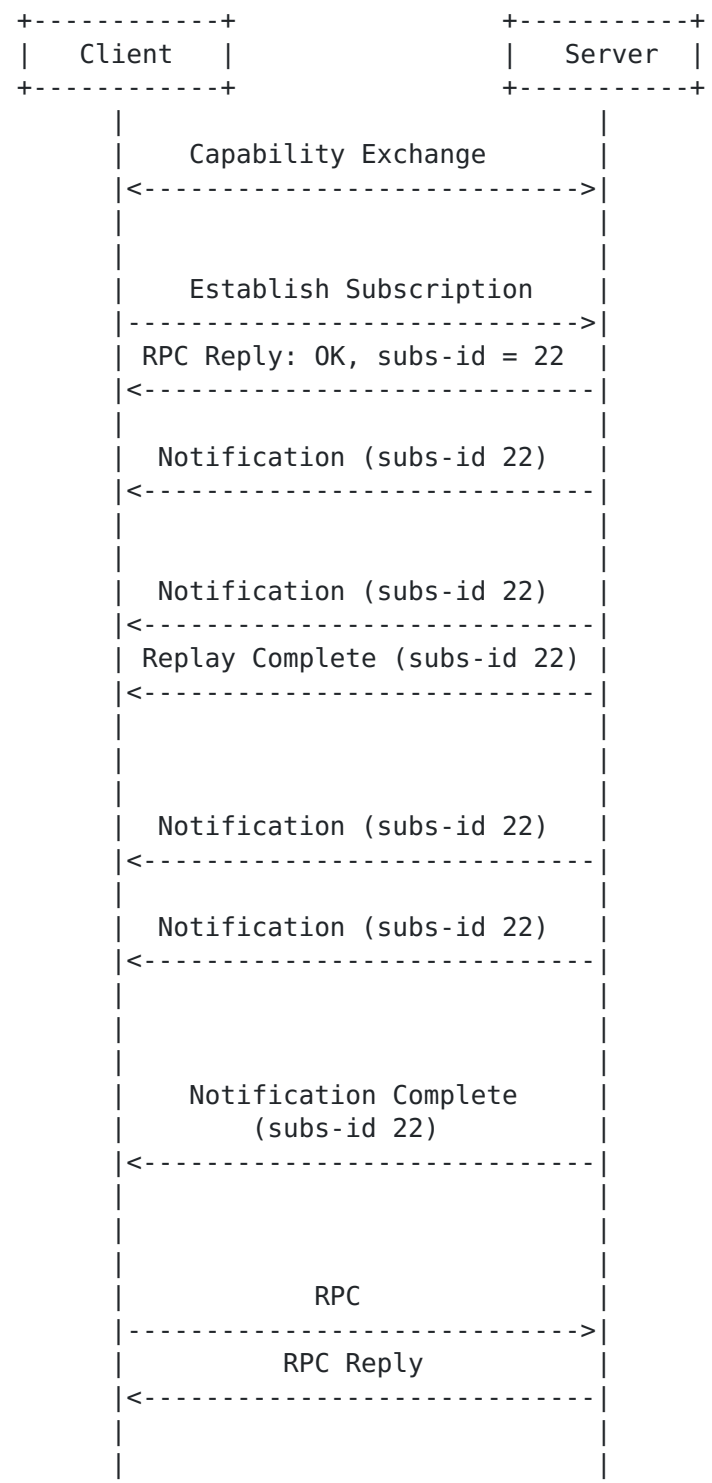


Figure 43: notificationComplete notification

2.10.3. subscription-started

```
<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-event-notifications"/>
    <subscription-id>52</subscription-id>
    <filter netconf:type="xpath"
              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')]"/>
  </subscription-started/>
</notification>
```

Figure 44: subscription-started control plane notification

The equivalent using JSON encoding would be:

```
<notification
       xmlns=" urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2007-09-01T10:00:00Z</eventTime>
  <notification-contents-json>
       {
          "notif-bis:subscription-started": {
              "subscription-id" : 52
              ((Open Item: express filter in json))
          }
        }
  </notification-contents-json>
</notification>
```

Figure 45: subscription-started control plane notification (JSON encoding)

2.10.4. subscription-modified

```
<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-event-notifications"/>
    <subscription-id>52</subscription-id>
    <filter netconf:type="xpath"
              xmlns:ex="http://example.com/event/1.0"
              select="/ex:event[ex:eventClass='fault']"/>
  </subscription-modified/>
</notification>
```

Figure 46: subscription-modified control plane notification

2.10.5. subscription-terminated

```
<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-event-notifications"/>
    <subscription-id>52</subscription-id>
    <reason>subscription-deleted</reason>
  </subscription-terminated/>
</notification>
```

Figure 47: subscription-terminated control plane notification

2.10.6. subscription-suspended

```
<notification
    xmlns=" urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2007-09-01T10:00:00Z</eventTime>
  <subscription-suspended
        xmlns="urn:ietf:params:xml:ns:yang:ietf-event-notifications"/>
    <subscription-id>52</subscription-id>
    <reason>internal-error</reason>
  </subscription-suspended/>
</notification>
```

Figure 48: subscription-suspended control plane notification

2.10.7. subscription-resumed

```
  <notification
      xmlns=" urn:ietf:params:xml:ns:netconf:notification:1.0">
    <eventTime>2007-09-01T10:00:00Z</eventTime>
    <subscription-resumed
          xmlns="urn:ietf:params:xml:ns:yang:ietf-event-notifications"/>
      <subscription-id>52</subscription-id>
      <reason>internal-error</reason>
    </subscription-resumed/>
  </notification>
```

Figure 49: subscription-resumed control plane notification

2.10.7.1. Message Flow Examples

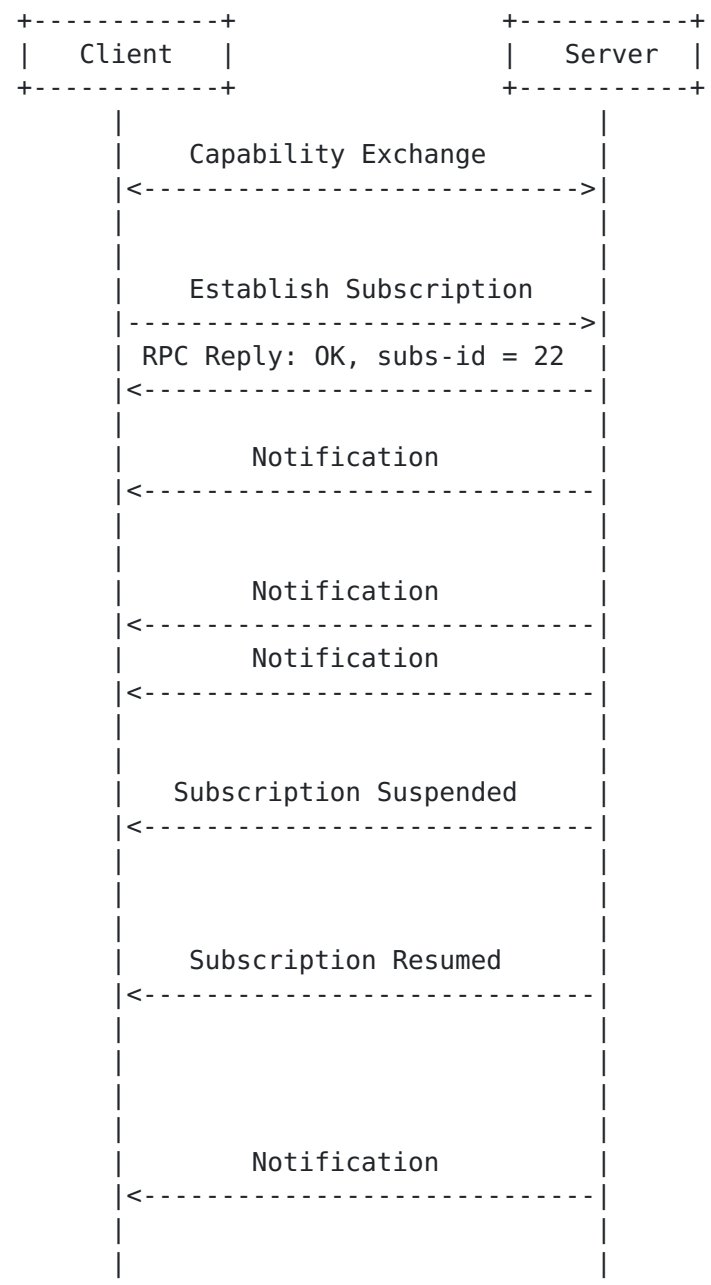
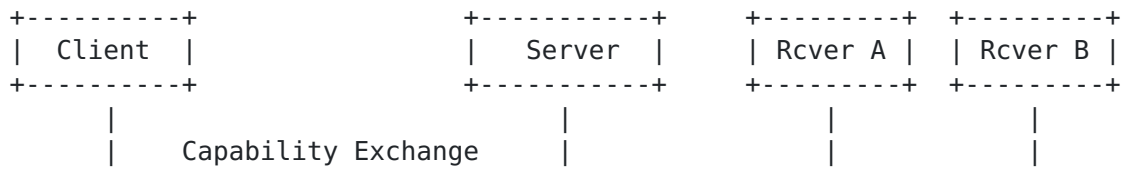


Figure 50: subscription-suspended and Resumed Notifications



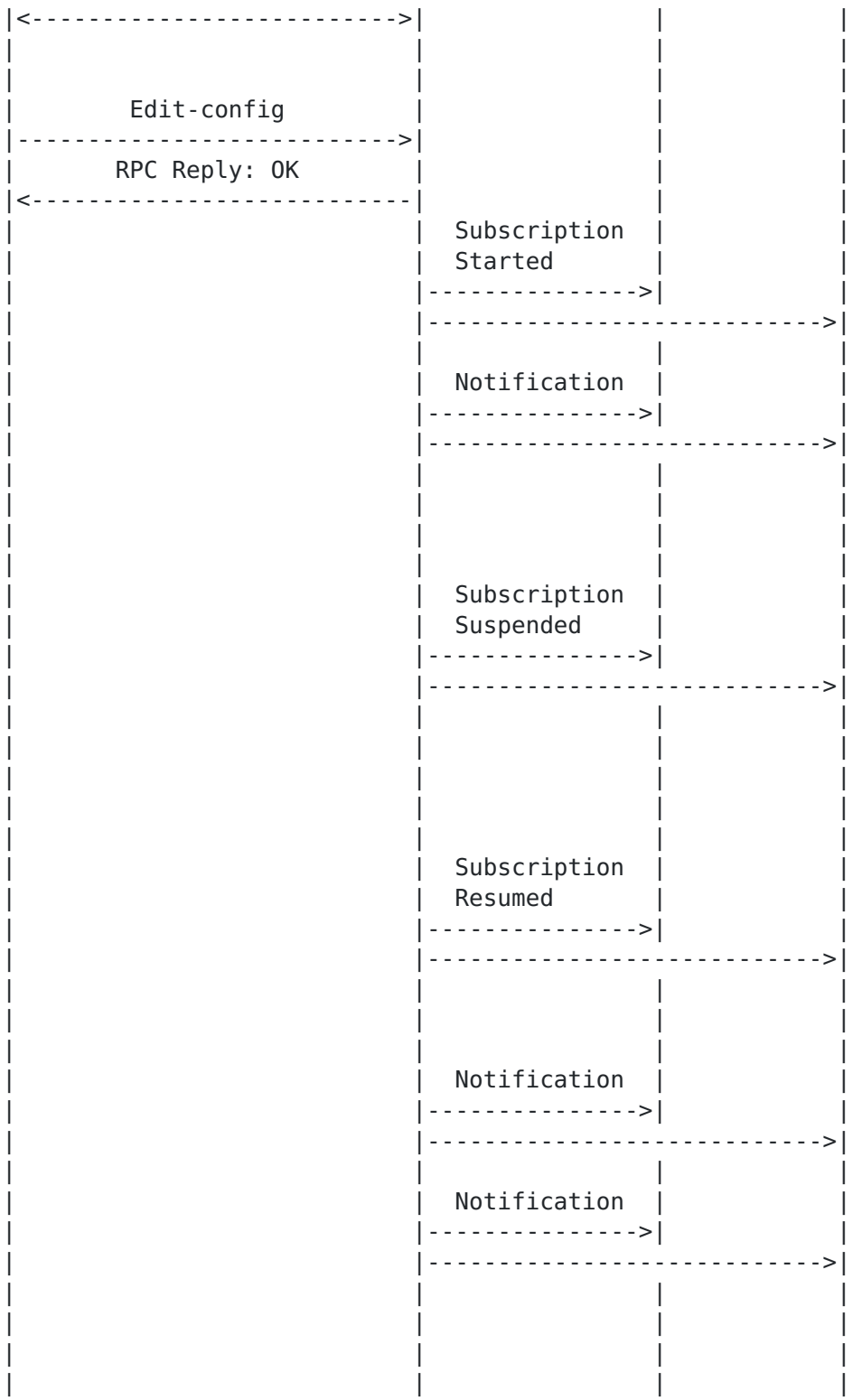

Edit-config
RPC Reply: OK
Subscription Started
Notification
Subscription Suspended
Subscription Resumed
Notification
Notification

Figure 51: subscription-suspended and subscription-resumed notifications (configured subscriptions)

3. Backwards Compatibility

3.1. Capabilities

Capabilities are advertised in messages sent by each peer during session establishment [RFC6241]. Servers supporting the features in this document must advertise both capabilities "urn:ietf:params:netconf:capability:notification:1.0" and "urn:ietf:params:netconf:capability:notification:1.1".

An example of a hello message by a server during session establishment would be:

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>
      urn:ietf:params:xml:ns:netconf:base:1.0
    </capability>
    <capability>
      urn:ietf:params:netconf:capability:startup:1.0
    </capability>
    <capability>
      urn:ietf:params:netconf:capability:notification:1.0
    </capability>
    <capability>
      urn:ietf:params:netconf:capability:notification:1.1
    </capability>
  </capabilities>
  <session-id>4</session-id>
</hello>
```

Figure 52: Hello message

Clients that only support [RFC5277] recognize capability "urn:ietf:params:netconf:capability:notification:1.0" and ignore capability "urn:ietf:params:netconf:capability:notification:1.1". This allows them interacting with the server as per [RFC5277]. Clients that support the features in this document recognize both capabilities. This allows them interacting with the server as per this document.

3.2. Stream Discovery

In order to maintain backwards compatibility, clients that only support [RFC5277] can retrieve the list of available event streams executing a <get> operation against the container "/netconf/streams".

4. Security Considerations

The security considerations from the base NETCONF document [RFC6241] also apply to the notification capability.

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 NETCONF content involved. When a <get> is received that refers to the content defined in this memo, clients should only be able to view the content for which they have sufficient privileges. <create-subscriptiont> and <establish-subscriptiont> operations can be considered like deferred <get>, and the content that different users can access may vary. This different access is reflected in the <notificationt> that different users are able to subscribe to.

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 <create-subscription> requests, then these subscriptions accumulate and may use up system resources. In such a situation, subscriptions can be terminated by terminating the suspect underlying NETCONF sessions using the <kill-session> operation. If the client uses <establish-subscription>, the server can also suspend or terminate subscriptions with per-subscription granularity.

A subscription could be configured on another receiver's behalf, with the goal of flooding that receiver with updates. One or more publishers could be used to overwhelm a receiver, which doesn't even support subscriptions. Clients that do not want pushed data need only terminate or refuse any transport sessions from the publisher. In addition, the NETCONF Authorization Control Model [RFC6536] SHOULD be used to control and restrict authorization of subscription configuration. This control models permits specifying per-user

permissions to receive specific event notification types. The permissions are specified as a set of access control rules.

Note that streams can define additional authorization requirements. For instance, in [netconf-yang-push] each of the elements in its data plane notifications must also go through access control.

5. Acknowledgments

We wish to acknowledge the helpful contributions, comments, and suggestions that were received from: Andy Bierman, Yan Gang, Peipei Guo, Susan Hares, Tim Jenkins, Balazs Lengyel, Kent Watsen, and Guangying Zheng.

6. References

6.1. 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, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <http://www.rfc-editor.org/info/rfc3339>.

[RFC5277] Chisholm, S. and H. Trevino, "NETCONF Event Notifications", RFC 5277, DOI 10.17487/RFC5277, July 2008, <http://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, <http://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, <http://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, <http://www.rfc-editor.org/info/rfc6536>.

6.2. Informative References

[event-notifications]
Gonzalez Prieto, A., Clemm, A., Voit, Eric., Nilsen-Nygaard, E., Tripathy, A., Chisholm, S., and H. Trevino, "Subscribing to YANG-Defined Event Notifications", June 2016, <https://datatracker.ietf.org/doc/draft-gonzalez-netconf-5277bis/>.

[netconf-yang-push]
Clemm, A., Gonzalez Prieto, A., Voit, Eric., Tripathy, A., and E. Nilsen-Nygaard, "Subscribing to YANG datastore push updates", February 2016, <https://datatracker.ietf.org/doc/draft-ietf-netconf-yang-push/>.

Appendix A. Issues being worked and resolved

(To be removed by RFC editor prior to publication)

A.1. Unresolved Issues

NT1 - Support multiple create-subscriptions over a single NETCONF session? or only multiple establish-subscription? Recommend only establish-subscription anytime there may be more than one subscription, otherwise variations of supported RFCs and error states proliferate.

NT2 - Configured subscription need to be refined in [event-notifications] and then adjust this document based on it.

NT3 - Express filter in JSON should be documented.

A.2. Agreement in principal

NT1 - When a list of requirements for secure transport is provided via a Configured Subscription, this list must be met via Netconf transport. Specifics still are needed for draft on the "how".

NT2 - Along with rfc5277bis, this draft to obsolete 5277

NT3 - Stream discovery. Adopted mechanism in 5277-bis and include backwards comptability support.

Authors' Addresses

Alberto Gonzalez Prieto
Cisco Systems

Email: albertgo@cisco.com

Alexander Clemm
Cisco Systems

Email: alex@cisco.com

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

Sharon Chisholm
Ciena

Email: schishol@ciena.com

Hector Trevino
Cisco Systems

Email: htrevino@cisco.com