Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 30, 2008

S. Chisholm
Nortel
H. Trevino
Cisco

May 29, 2008

NETCONF Event Notifications
draft-ietf-netconf-notification-13.txt

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on November 30, 2008.

Copyright Notice

Copyright (C) The IETF Trust (2008).

Abstract

This document defines mechanisms that provide an asynchronous message notification delivery service for the NETCONF protocol. This is an optional capability built on top of the base NETCONF definition. This document defines the capabilities and operations necessary to support this service.

Table of Contents

1. Introduction 4
1.1. Definition of Terms 4
1.2. Motivation 5
1.3. Event Notifications in NETCONF 6
2. Notification-Related Operations 7
2.1. Subscribing to Receive Event Notifications 7
2.1.1. <create-subscription> 7
2.2. Sending Event Notifications 10
2.2.1. <notification> 10
2.3. Terminating the Subscription 10
3. Supporting Concepts 11
3.1. Capabilities Exchange 11
3.1.1. Capability Identifier 11
3.1.2. Capability Example 11
3.2. Event Streams 11
3.2.1. Event Stream Definition 13
3.2.2. Event Stream Content Format 13
3.2.3. Default Event Stream 13
3.2.4. Event Stream Sources 13
3.2.5. Event Stream Discovery 13
3.3. Notification Replay 16
3.3.1. Overview 16
3.3.2. Creating a Subscription with Replay 17
3.4. Notification Management Schema 17
3.5. Subscriptions Data 21
3.6. Filter Mechanics 21
3.6.1. Filtering 21
3.7. Message Flow 21
4. XML Schema for Event Notifications 24
5. Filtering Examples 28
5.1. Subtree Filtering 31
5.2. XPATH filters 32
6. Interleave Capability 34
6.1. Description 34
6.2. Dependencies 34
6.3. Capability Identifier 34
6.4. New Operations 34

6.5. Modifications to Existing Operations 34
7. Security Considerations 35
8. IANA Considerations 37
9. Acknowledgements 38
10. Normative References 39
Appendix A. Change Log 40
A.1. Version -08 40
A.2. Version -09 42
A.3. Version -10 44
A.4. Version -11 44
A.5. Version -12 45
A.6. Version -13 45
Authors' Addresses 48
Intellectual Property and Copyright Statements 49

1. Introduction

[NETCONF] can be conceptually partitioned into four layers:

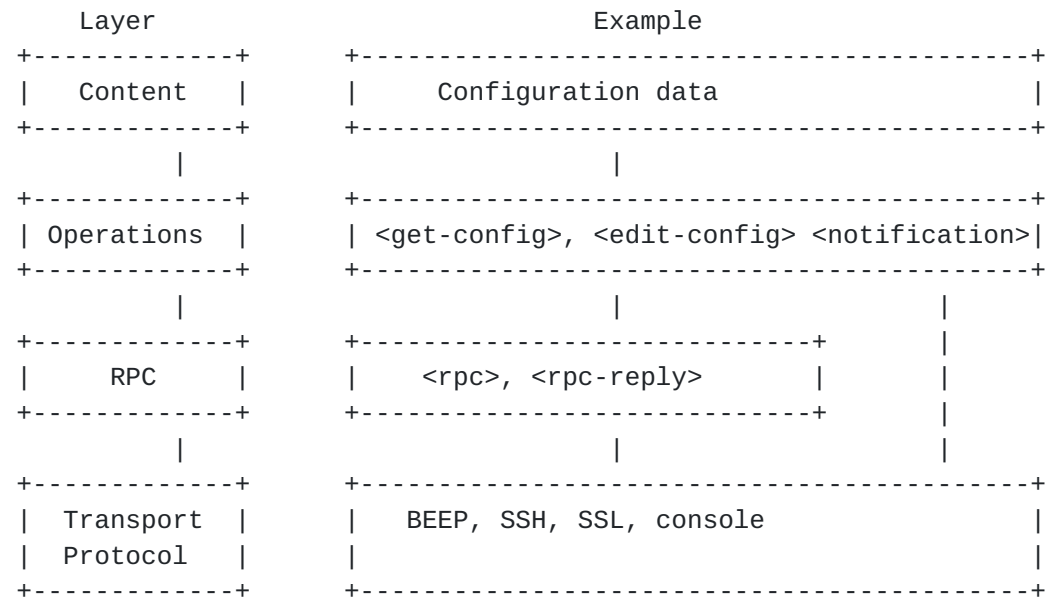


Figure 1

This document defines mechanisms which provide an asynchronous message notification delivery service for the [NETCONF] protocol. This is an optional capability built on top of the base NETCONF definition. This memo defines the capabilities and operations necessary to support this service.

1.1. Definition of Terms

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 [RFC2119].

Element: An [XML] Element.

Subscription: An agreement and method to receive event notifications over a NETCONF session. A concept related to the delivery of notifications (if there are any to send) involving destination and selection of notifications. It is bound to the lifetime of a session.

Operation: This term is used to refer to NETCONF protocol operations [NETCONF]. Within this document, operation refers to NETCONF protocol operations defined in support of NETCONF notifications.

Event: An event is something that happens which may be of interest - a configuration change, a fault, a change in status, crossing a threshold, or an external input to the system, for example. Often this results in an asynchronous message, sometimes referred to as a notification or event notification, being sent to interested parties to notify them that this event has occurred.

Replay: The ability to send/re-send previously logged notifications upon request. These notifications are sent asynchronously. This feature is implemented by the NETCONF server and invoked by the NETCONF client.

Stream: An event stream is a set of event notifications matching some forwarding criteria and available to NETCONF clients for subscription.

Filter: A parameter that indicates which subset of all possible events are of interest. A filter is defined as one or more filter element [NETCONF], which each identifies a portion of the overall filter.

1.2. Motivation

The motivation for this work is to enable the sending of asynchronous messages that are consistent with the data model (content) and security model used within a NETCONF implementation.

The scope of the work aims meeting the following operational needs:

- Initial release should ensure it supports notifications in support of configuration operations.
- It should be possible to use the same data model for notifications as for configuration operations.
- Solution should support a reasonable message size limit (i.e., not too short)
- The notifications should be carried over a connection-oriented delivery mechanism.
- A subscription mechanism for notifications should be provided. This takes into account that a NETCONF server does not send notifications before being asked to do so and that it is the

NETCONF client who initiates the flow of notifications.

- A filtering mechanism for sending notifications should be put in place within the NETCONF server.

- The information contained in a notification should be sufficient so that it can be analyzed independent of the transport mechanism. In other words the data content fully describes a notification; protocol information is not needed to understand a notification.

- The server should have the capability to replay locally logged notifications.

1.3. Event Notifications in NETCONF

This memo defines a mechanism whereby the NETCONF client indicates interest in receiving event notifications from a NETCONF server by creating a subscription to receive event notifications. The NETCONF server replies to indicate whether the subscription request was successful and, if it was successful, begins sending the event notifications to the NETCONF client as the events occur within the system. These event notifications will continue to be sent until either the NETCONF session is terminated or the subscription terminates for some other reason. The event notification subscription allows a number of options to enable the NETCONF client to specify which events are of interest. These are specified when the subscription is created. Note that a subscription cannot be modified once created.

The NETCONF server MUST accept and process the <close-session> operation, even while the notification subscription is active. The NETCONF server MAY accept and process other commands, otherwise they will be rejected and the server MUST send a 'resource-denied' error. A NETCONF server advertises support of the ability to process other commands via the interleave capability.

2. Notification-Related Operations

2.1. Subscribing to Receive Event Notifications

The event notification subscription is initiated by the NETCONF client and responded to by the NETCONF server. A subscription is bound to a single stream for the lifetime of the subscription. When the event notification subscription is created, the events of interest are specified.

Content for an event notification subscription can be selected by applying user-specified filters.

2.1.1. <create-subscription>

Description:

This operation initiates an event notification subscription which will send asynchronous event notifications to the initiator of the command until the subscription terminates.

Parameters:

Stream:

An optional parameter, <stream>, that indicates which stream of events is of interest. If not present, events in the default NETCONF stream will be sent.

Filter:

An optional parameter, <filter>, that indicates which subset of all possible events is of interest. The format of this parameter is the same as that of the filter parameter in the NETCONF protocol operations. If not present, all events not precluded by other parameters will be sent. See section 3.6 for more information on filters.

Start Time:

A parameter, <startTime>, used to trigger the replay feature and indicate that the replay should start at the time specified. If <startTime> is not present, this is not a replay subscription. It is not valid to specify start times that are later than the current time. If the <startTime> specified is earlier than the log can support, the replay will begin with the earliest available notification. This parameter is of type dateTime and compliant to [RFC3339]. Implementations must

support time zones.

Stop Time:

An optional parameter, <stopTime>, used with the optional replay feature to indicate the newest notifications of interest. If stop time is not present, the notifications will continue until the subscription is terminated. Must be used with and be later than <startTime>. Values of <stopTime> in the future are valid. This parameter is of type dateTime and compliant to [RFC3339]. Implementations must support time zones.

Positive Response:

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

Negative Response:

An <rpc-error> element is included within the <rpc-reply> if the request cannot be completed for any reason. Subscription requests will fail 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.

If the optional replay feature is requested but it is not supported by the NETCONF server, 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

If a <stopTime> is requested which is earlier then 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.

If a <startTime> is requested which is later then 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.

2.1.1.1. Usage Example

The following demonstrates creating a simple subscription. More complex examples can be found in section 5.

```
<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>
```

2.2. Sending Event Notifications

Once the subscription has been set up, the NETCONF server sends the event notifications asynchronously over the connection.

2.2.1. <notification>

Description:

An event notification is sent to the client who initiated a <create-subscription> command asynchronously 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 an RPC method but rather the top level element identifying the one-way message as a notification.

Parameters:

eventTime

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.

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.

Response:

No response. Not applicable.

2.3. Terminating the Subscription

Closing of the event notification subscription can be done by using the <close-session> operation from the subscriptions session or terminating the NETCONF session (<kill-session>) or the underlying transport session from another session. If a stop time is provided when the subscription is created, the subscription will terminate after the stop time is reached. In this case, the NETCONF session will still be an active session.

3. Supporting Concepts

3.1. Capabilities Exchange

The ability to process and send event notifications is advertised during the capability exchange between the NETCONF client and server.

3.1.1. Capability Identifier

"urn:ietf:params:netconf:capability:notification:1.0"

3.1.2. Capability Example

```
<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>
    </capabilities>
  <session-id>4</session-id>
</hello>
```

3.2. Event Streams

An event stream is defined as a set of event notifications matching some forwarding criteria.

Figure 2 illustrates the notification flow and concepts identified in this document. It does not mandate and/or preclude an implementation. The following is observed from the diagram below: System components (c1..cn) generate event notifications which are passed to a central component for classification and distribution. The central component inspects each event notification and matches the event notification against the set of stream definitions. When a match occurs, the event notification is considered to be a member of that event stream (stream 1..stream n). An event notification may be part of multiple event streams.

At some point after the NETCONF server receives the internal event from a stream, it is converted to an appropriate XML encoding by the

server, and a <notification> element is ready to send to all NETCONF sessions subscribed to that stream.

After generation of the <notification> element, access control is applied by the server. If a session does not have permission to receive the <notification>, then it is discarded for that session, and processing of the internal event is completed for that session.

When a NETCONF client subscribes to a given event stream, user-defined filter elements, if applicable, are applied to the event stream and matching event notifications are forwarded to the NETCONF server for distribution to subscribed NETCONF clients. A filter is transferred from the client to the server during the <create-subscription> operation and applied against each <notification> element generated by the stream. For more information on filtering, see section 3.6.

A notification logging service may also be available, in which case, the central component logs notifications. The NETCONF server may later retrieve logged notifications via the optional replay feature. For more information on replay, see section 3.3.

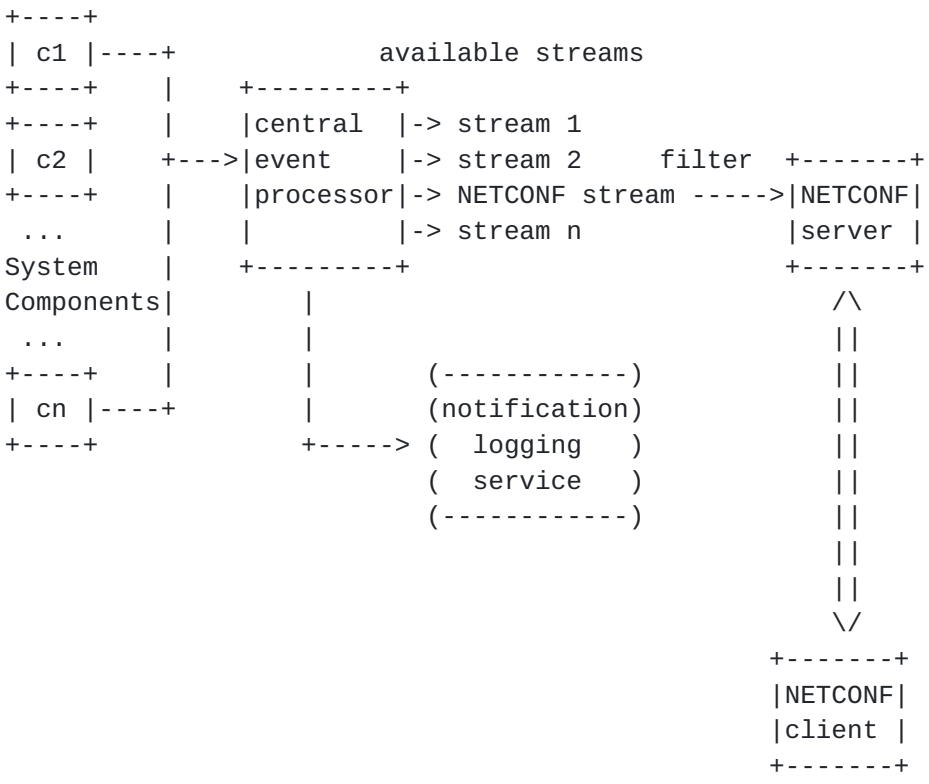


Figure 2

3.2.1. Event Stream Definition

Event streams are predefined on the managed device. The configuration of event streams is outside the scope of this document. However, it is envisioned that event streams are either pre-established by the vendor (pre-configured), user configurable (e.g., part of the device's configuration) or both. Device vendors may allow event stream configuration via the NETCONF protocol (i.e., <edit-config> operation).

3.2.2. Event Stream Content Format

The contents of all event streams made available to a NETCONF client (i.e., the notification sent by the NETCONF server) MUST be encoded in XML.

3.2.3. 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. The exact string "NETCONF" is used during advertisement of stream support during the <get> operation on <streams> and during the <create-subscription> operation. Definition of the event notifications and their contents, beyond the inclusion of <eventTime>, for this event stream is outside the scope of this document.

3.2.4. Event Stream Sources

With the exception of the default event stream (NETCONF), specification of additional event stream sources (e.g., SNMP, syslog) is outside the scope of this document. NETCONF server implementations may leverage any desired event stream source in the creation of supported event streams.

3.2.5. Event Stream Discovery

A NETCONF client retrieves the list of supported event streams from a NETCONF server using the <get> operation.

3.2.5.1. Name Retrieval using <get> operation

The list of available event streams is retrieved by requesting the <streams> subtree via a <get> operation. Available event streams for the requesting session are returned in the reply containing the

<name> and <description> elements, where the <name> element is mandatory, and its value is unique within the scope of a NETCONF server. An empty reply is returned if there are no available event streams, due to user-specified filters on the <get> operation .

Additional information available about a stream include whether notification replay is available and if so, the timestamp of the earliest possible notification to replay.

The following example shows retrieving the list of available event stream list using the <get> operation.

```
<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>
```

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

```
<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>
             2007-07-08T00:00:00Z
           </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>
```

3.2.5.2. Event Stream Subscription

A NETCONF client may request from the NETCONF server the list of event streams available to this session and then issue a <create-subscription> request with the desired event stream name. Omitting the event stream name from the <create-subscription> request results in subscription to the default NETCONF event stream.

3.2.5.2.1. Filtering Event Stream Contents

The set of event notifications delivered in an event stream may be further refined by applying a user-specified filter supplied at subscription creation time (<create-subscription>). This is a transient filter associated with the event notification subscription and does not modify the event stream configuration. The filter element is applied against the contents of the <notification> wrapper and not the wrapper itself. See section 5 for examples. Either subtree or XPATH filtering can be used.

XPATH support for the Notification capability is advertised as part of the normal XPATH capability advertisement. If XPATH support is advertised via the XPATH capability then XPATH is supported for notification filtering and if this capability is not advertised, XPATH is not supported for notification filtering.

3.3. Notification Replay

3.3.1. Overview

Replay is the ability to create an event subscription that will resend recently generated notifications, or in some cases send them for the first time to a particular NETCONF client. These notifications are sent the same way as normal notifications.

A replay of notifications is specified by including the optional <startTime> parameter to the subscription command, which indicates the start time of the replay. The end time is specified using the optional <stopTime> parameter. If not present, notifications will continue to be sent until the subscription is terminated.

A notification stream that supports replay is not expected to have an unlimited supply of saved notifications available to accommodate any replay request. Clients can query <replayLogCreationTime> and <replayLogAgedTime> to learn about the availability of notifications for replay.

The actual number of stored notifications available for retrieval at any given time is a NETCONF server implementation specific matter. Control parameters for this aspect of the feature are outside the scope of this document.

Replay is dependent on a notification stream supporting some form of notification logging, although it puts no restrictions on the size or form of the log, or where it resides within the device. Whether or not a stream supports replay can be discovered by doing a <get> operation on the <streams> element of the Notification Management

Schema and looking at the value of the <replaySupport> object. This schema also provides the <replayLogCreationTime> element to indicate the earliest available logged notification.

3.3.2. Creating a Subscription with Replay

This feature uses optional parameters to the <create-subscription> command called <startTime> and <stopTime>. <startTime> identifies the earliest date and time of interest for event notifications being replayed and also indicates that a subscription will be providing replay of notifications. Events generated before this time are not matched. <stopTime> specifies the latest date and time of interest for event notifications being replayed. If it is not present, then notifications will continue to be sent until the subscription is terminated.

Note that <startTime> and <stopTime> are associated with the time an event was generated by the event source.

A <replayComplete> notification is sent to indicate that all of the replay notifications have been sent and must not be sent for any other reason. If this subscription has a stop time, then this session becomes a normal NETCONF session again. The NETCONF server will then accept <rpc> operations even if the server did not previously accept such operations due to lack of interleave support. In the case of a subscription without a stop time, after the <replayComplete> notification has been sent, it can be expected that any notifications generated since the start of the subscription creation will be sent, followed by notifications as they arise naturally within the system.

The <replayComplete> and <notificationComplete> notifications cannot be filtered out. They will always be sent on a replay subscription that specified a startTime and stopTime respectively.

3.4. Notification Management Schema

This Schema is used to learn about the event streams supported on the system. It also contains the definition of the <replayComplete> and <notificationComplete> notifications, which are sent to indicate that an event replay has sent all applicable notifications and that the subscription has terminated, respectively.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0"
    xmlns:ncEvent="urn:ietf:params:xml:ns:netconf:notification:1.0"
```

```
    xmlns:manageEvent="urn:ietf:params:xml:ns:netmod:notification"
    targetNamespace="urn:ietf:params:xml:ns:netmod:notification"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified"
    xml:lang="en" version="1.0">
    <xs:annotation>
        <xs:documentation xml:lang="en">
            A schema that can be used to learn about current
            event streams. It also contains the replayComplete
            and notificationComplete  notification.
        </xs:documentation>
    </xs:annotation>


<xs:import namespace="http://www.w3.org/XML/1998/namespace"
        schemaLocation="http://www.w3.org/2001/xml.xsd"/>
<xs:import namespace="urn:ietf:params:xml:ns:netconf:base:1.0"
    schemaLocation="netconf.xsd"/>
<xs:import namespace=
    "urn:ietf:params:xml:ns:netconf:notification:1.0"
      schemaLocation="notification.xsd"/>
<!-- The above  schemaLocation value is a placeholder and the actual
                                      value will be assigned by IANA -->

<xs:element name="netconf" type="manageEvent:Netconf"/>

<xs:complexType name="Netconf">
  <xs:sequence>
      <xs:element name="streams" >
        <xs:annotation>
           <xs:documentation>
             The list of event streams supported by the
             system. When a query is issued, the returned
             set of streams is determined based on user
             privileges.
           </xs:documentation>
         </xs:annotation>
         <xs:complexType>
           <xs:sequence minOccurs="1" maxOccurs="unbounded">
             <xs:element name="stream">
                <xs:annotation>
                  <xs:documentation>
                    Stream name, description and other information.
                  </xs:documentation>
                </xs:annotation>
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="name"
```

```
            type="ncEvent:streamNameType">
          <xs:annotation>
            <xs:documentation>
              The name of the event stream. If this is
              the default NETCONF stream, this must have
              the value "NETCONF".
            </xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="description"
                          type="xs:string">
          <xs:annotation>
            <xs:documentation>
              A description of the event stream, including
              such information as the type of events that
              are sent over this stream.
            </xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="replaySupport"
                          type="xs:boolean">
         <xs:annotation>
            <xs:documentation>
              An indication of whether or not event replay
              is available on this stream.
            </xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="replayLogCreationTime"
                       type="xs:dateTime" minOccurs="0">
          <xs:annotation>
            <xs:documentation>
           The timestamp of the creation of the log
           used to support the replay function on
           this stream.
           Note that this might be earlier then
           the earliest available
           notification in the log. This object
           is updated if the log resets
           for some reason. This
           object MUST be present if replay is
           supported.
            </xs:documentation>
          </xs:annotation>
         </xs:element>
         <xs:element name="replayLogAgedTime"
                type="xs:dateTime" minOccurs="0">
          <xs:annotation>
```

```
                        <xs:documentation>
                          The timestamp of the last notification
                          aged out of the log. This
                          object MUST be present if replay is
                          supported and any notifications
                          have been aged out of the log.
                        </xs:documentation>
                      </xs:annotation>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
    </xs:sequence>
    </xs:complexType>

    <xs:complexType name="ReplayCompleteNotificationType">
        <xs:complexContent>
            <xs:extension base="ncEvent:NotificationContentType"/>
        </xs:complexContent>
    </xs:complexType>

    <xs:element name="replayComplete"
        type="manageEvent:ReplayCompleteNotificationType"
        substitutionGroup="ncEvent:notificationContent">
                <xs:annotation>
          <xs:documentation>
            This notification is sent to signal the end of a replay
            portion of a subscription.
          </xs:documentation>
        </xs:annotation>
        </xs:element>

    <xs:complexType name="NotificationCompleteNotificationType">
        <xs:complexContent>
            <xs:extension base="ncEvent:NotificationContentType"/>
        </xs:complexContent>
    </xs:complexType>

    <xs:element name="notificationComplete"
        type="manageEvent:NotificationCompleteNotificationType"
        substitutionGroup="ncEvent:notificationContent">
                <xs:annotation>
          <xs:documentation>
            This notification is sent to signal the end of a
            notification subscription. It is sent in the case
```

```
            that stopTime was specified during the creation of
            the subscription.
          </xs:documentation>
        </xs:annotation>
        </xs:element>

</xs:schema>
```

3.5. Subscriptions Data

Subscriptions are non-persistent state information and their lifetime is defined by their session or by the <stopTime> parameter.

3.6. Filter Mechanics

If a filter element is specified to look for data of a particular value, and the data item is not present within a particular event notification for its value to be checked against, the notification will be filtered out. For example, if one were to check for 'severity=critical' in a configuration event notification where this field was not supported, then the notification would be filtered out.

For subtree filtering, a non-empty node set means that the filter matches. For XPath filtering, the mechanisms defined in [XPATH] should be used to convert the returned value to boolean.

3.6.1. Filtering

Filtering is explicitly stated when the event notification subscription is created. This is specified via the 'filter' parameter. A Filter only exists as a parameter to the subscription.

3.7. Message Flow

The following figure depicts message flow between a NETCONF client (C) and NETCONF server (S) in order to create a subscription and begin the flow of notifications. This subscription specified a <startTime>, so the server starts by replaying logged notifications. It is possible that many rpc/rpc-reply sequences occur before the subscription is created, but this is not depicted in the figure.

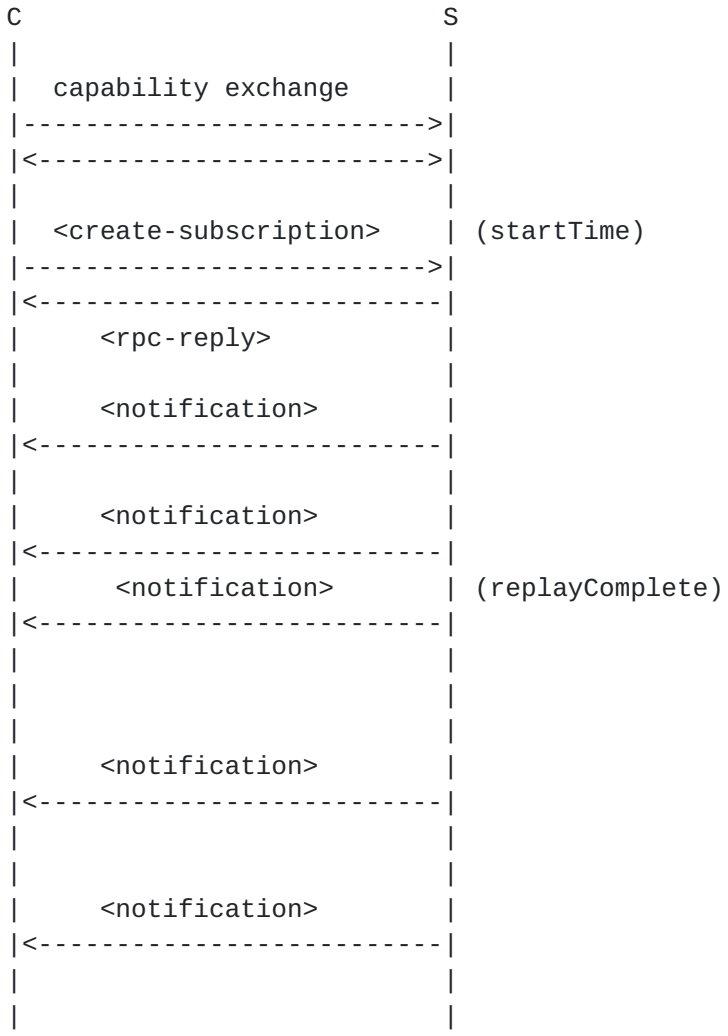


Figure 3

The following figure depicts message flow between a NETCONF client (C) and NETCONF server (S) in order to create a subscription and begin the flow of notifications. This subscription specified a <startTime> and <stopTime> so it starts by replaying logged notifications and then returns to be a normal command-response NETCONF session after the <replayComplete> and <notificationComplete> notifications are sent and it is available to process <rpc> requests. It is possible that many rpc/rpc-reply sequences occur before the subscription is created, but this is not depicted in the figure.

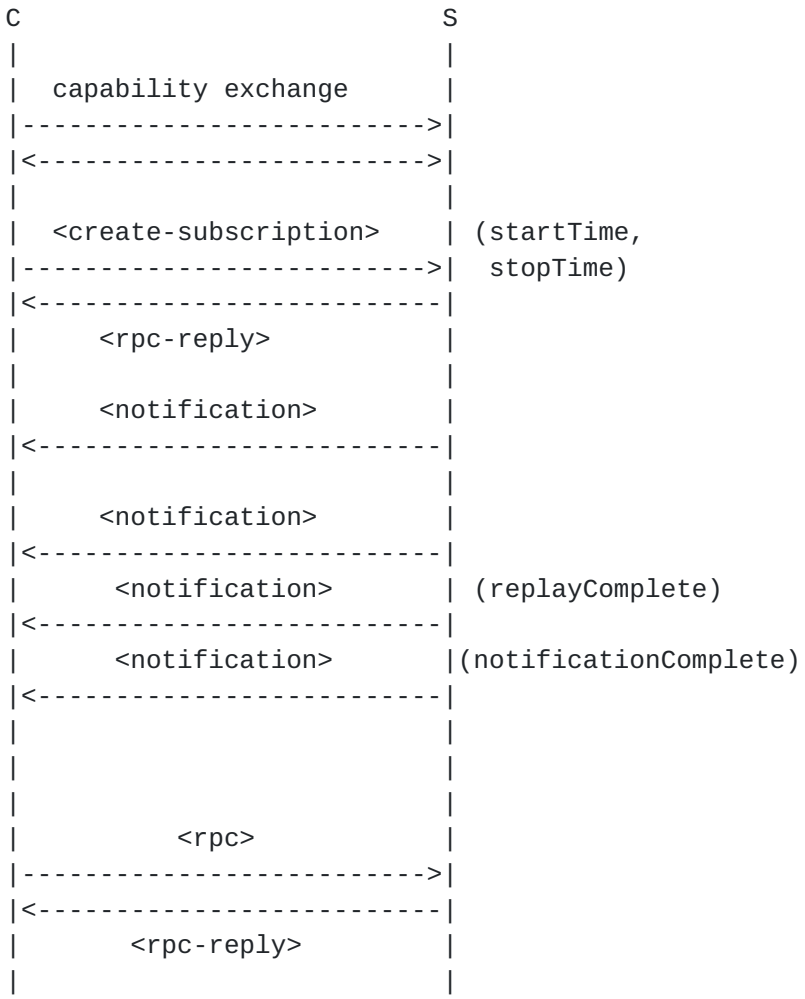


Figure 4

4. XML Schema for Event Notifications

The following [XML Schema] defines NETCONF Event Notifications.

```
<?xml version="1.0" encoding="UTF-8"?>
  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
     xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0"
     xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0"
     targetNamespace=
        "urn:ietf:params:xml:ns:netconf:notification:1.0"
     elementFormDefault="qualified"
     attributeFormDefault="unqualified"
       xml:lang="en">

    <!-- import standard XML definitions -->

     <xs:import namespace="http://www.w3.org/XML/1998/namespace"
                schemaLocation="http://www.w3.org/2001/xml.xsd">
       <xs:annotation>
         <xs:documentation>
           This import accesses the xml: attribute groups for the
           xml:lang as declared on the error-message element.
         </xs:documentation>
       </xs:annotation>
     </xs:import>

     <!-- import base netconf definitions -->
     <xs:import namespace="urn:ietf:params:xml:ns:netconf:base:1.0"
       schemaLocation="netconf.xsd"/>



<!-- ************** Symmetrical Operations  ********************-->

     <!-- <create-subscription> operation -->


    <xs:complexType name="createSubscriptionType">
        <xs:complexContent>
            <xs:extension base="netconf:rpcOperationType">
                <xs:sequence>
                    <xs:element name="stream"
                        type="streamNameType" minOccurs="0">
                        <xs:annotation>
                            <xs:documentation>
                               An optional parameter that indicates
                               which stream of events is of interest. If
```

```
                        not present, then events in the default
                        NETCONF stream will be sent.
                    </xs:documentation>
                </xs:annotation>
            </xs:element>
                <xs:element name="filter"
                    type="netconf:filterInlineType"
                    minOccurs="0">
                    <xs:annotation>
                        <xs:documentation>
                            An optional parameter that indicates
                            which subset of all possible events
                            is of interest. The format of this
                            parameter is the same as that of the
                            filter parameter in the NETCONF
                            protocol operations. If not present,
                            all events not precluded by other
                            parameters will be sent.
                        </xs:documentation>
                    </xs:annotation>
                </xs:element>
            <xs:element name="startTime" type="xs:dateTime"
                minOccurs="0" >
                <xs:annotation>
                    <xs:documentation>
                        A parameter used to trigger the replay
                        feature and indicates that the replay
                        should start at the time specified. If
                        start time is not present, this is not a
                        replay subscription.
                    </xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element name="stopTime" type="xs:dateTime"
                minOccurs="0" >
                <xs:annotation>
                    <xs:documentation>
                        An optional parameter used with the
                        optional replay feature to indicate the
                        newest notifications of interest. If
                        stop time is not present, the
                        notifications will continue until the
                        subscription is terminated. Must be used
                        with startTime.
                    </xs:documentation>
                </xs:annotation>
            </xs:element>
        </xs:sequence>
```

```
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>


    <xs:simpleType name="streamNameType">
        <xs:annotation>
            <xs:documentation>
                The name of an event stream.
            </xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:string"/>
    </xs:simpleType>


    <xs:element name="create-subscription"
        type="createSubscriptionType"
        substitutionGroup="netconf:rpcOperation">
        <xs:annotation>
            <xs:documentation>
                The command to create a notification subscription. It
                takes as argument the name of the notification stream
                and filter. Both of those options
                limit the content of the subscription. In addition,
                there are two time-related parameters, startTime and
                stopTime, which can be used to select the time interval
                of interest to the notification replay feature.
            </xs:documentation>
        </xs:annotation>
    </xs:element>

<!-- ************** One-way Operations  ******************-->

     <!-- <Notification> operation -->
     <xs:complexType name="NotificationContentType"/>

    <xs:element name="notificationContent"
        type="NotificationContentType" abstract="true"/>

    <xs:complexType name="NotificationType">
        <xs:sequence>
            <xs:element name="eventTime" type="xs:dateTime">
              <xs:annotation>
                <xs:documentation>
                The time the event was generated by the event source
                </xs:documentation>
              </xs:annotation>
            </xs:element>
```

```
            <xs:element ref="notificationContent"/>
        </xs:sequence>
    </xs:complexType>

    <xs:element name="notification" type="NotificationType"/>

  </xs:schema>
```

5. Filtering Examples

The following section provides examples to illustrate the various methods of filtering content on an event notification subscription.

In order to illustrate the use of filter expressions, it is necessary to assume some of the event notification content. The examples below assume that the event notification schema definition has an <event> element at the top level consisting of the event class (e.g., fault, state, config), reporting entity and either severity or operational state.

Examples in this section are generated from the following fictional Schema.

```
 <?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://example.com/event/1.0"
    xmlns="http://example.com/event/1.0"
    elementFormDefault="qualified"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:ncEvent="urn:ietf:params:xml:ns:netconf:notification:1.0">

    <xs:import namespace=
        "urn:ietf:params:xml:ns:netconf:notification:1.0"
        schemaLocation="notification.xsd"/>


    <xs:complexType name="eventType">
        <xs:complexContent>
            <xs:extension base="ncEvent:NotificationContentType">
                <xs:sequence>
                    <xs:element name="eventClass" />
                    <xs:element name="reportingEntity">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:any namespace="##any"
                                processContents="lax"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                    <xs:choice>
                        <xs:element name="severity"/>
                        <xs:element name="operState"/>
                    </xs:choice>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:element name="event"
        type="eventType"
        substitutionGroup="ncEvent:notificationContent"/>


</xs:schema>
```

The above fictional notification definition could result in the following sample notification list, which is used in the examples in

this section.

```
<notification
   xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
   <eventTime>2007-07-08T00:01:00Z</eventTime>
   <event xmlns="http://example.com/event/1.0">
      <eventClass>fault</eventClass>
      <reportingEntity>
          <card>Ethernet0</card>
      </reportingEntity>
      <severity>major</severity>
    </event>
</notification>

<notification
  xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
   <eventTime>2007-07-08T00:02:00Z</eventTime>
   <event xmlns="http://example.com/event/1.0">
       <eventClass>fault</eventClass>
       <reportingEntity>
           <card>Ethernet2</card>
       </reportingEntity>
       <severity>critical</severity>
    </event>
</notification>

<notification
  xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
   <eventTime>2007-07-08T00:04:00Z</eventTime>
   <event xmlns="http://example.com/event/1.0">
       <eventClass>fault</eventClass>
       <reportingEntity>
            <card>ATM1</card>
        </reportingEntity>
        <severity>minor</severity>
   </event>
</notification>

<notification
  xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2007-07-08T00:10:00Z</eventTime>
  <event xmlns="http://example.com/event/1.0">
      <eventClass>state</eventClass>
      <reportingEntity>
          <card>Ethernet0</card>
      </reportingEntity>
      <operState>enabled</operState>
```

```
    </event>
  </notification>
```

5.1. Subtree Filtering

XML subtree filtering is not well-suited for creating elaborate filter definitions given that it only supports equality comparisons and application of the logical OR operators (e.g., in an event subtree give me all event notifications which have severity=critical or severity=major or severity=minor). Nevertheless, it may be used for defining simple event notification forwarding filters as shown below.

The following example illustrates how to select fault events which have severities of critical, major, or minor. The filtering criteria evaluation is as follows:

((fault & severity=critical) | (fault & severity=major) | (fault & severity=minor))

```
<netconf:rpc netconf: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">
    <filter netconf:type="subtree">
      <event xmlns="http://example.com/event/1.0">
        <eventClass>fault</eventClass>
        <severity>critical</severity>
      </event>
      <event xmlns="http://example.com/event/1.0">
        <eventClass>fault</eventClass>
        <severity>major</severity>
      </event>
      <event xmlns="http://example.com/event/1.0">
        <eventClass>fault</eventClass>
        <severity>minor</severity>
      </event>
    </filter>
  </create-subscription>
</netconf:rpc>
```

The following example illustrates how to select state or config EventClasses or fault events that are related to card Ethernet0. The filtering criteria evaluation is as follows:

```
   ( state | config | ( fault & ( card=Ethernet0)))
```

```
<netconf:rpc netconf: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">
        <filter netconf:type="subtree">
          <event xmlns="http://example.com/event/1.0">
            <eventClass>state</eventClass>
          </event>
          <event xmlns="http://example.com/event/1.0">
            <eventClass>config</eventClass>
          </event>
          <event xmlns="http://example.com/event/1.0">
            <eventClass>fault</eventClass>
            <reportingEntity>
              <card>Ethernet0</card>
            </reportingEntity>
          </event>
        </filter>
      </create-subscription>
</netconf:rpc>
```

5.2. XPATH filters

The following [XPATH] example illustrates how to select fault EventClass notifications that have severities of critical, major, or minor. The filtering criteria evaluation is as follows:

```
((fault) & ((severity=critical) | (severity=major) | (severity =
minor)))
```

```
   <netconf:rpc netconf: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">
       <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>
```

The following example illustrates how to select state and config EventClasses or fault events of any severity that come from card Ethernet0. The filtering criteria evaluation is as follows:

```
( state | config | (fault & card=Ethernet0))
```

```
<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">
    <filter netconf:type="xpath"
            xmlns:ex="http://example.com/event/1.0"
       select="/ex:event[
          (ex:eventClass='state' or ex:eventClass='config') or
          ((ex:eventClass='fault' and ex:card='Ethernet0'))]"/>
  </create-subscription>
</netconf:rpc>
```

6. Interleave Capability

6.1. Description

The Interleave capability indicates that the NETCONF peer supports the ability to interleave other NETCONF operations within a Notification subscription. This means the NETCONF server MUST receive, process and respond to NETCONF requests on a session with an active notification subscription. This capability helps scalability by reducing the total number of NETCONF sessions required by a given operator or management application.

6.2. Dependencies

This capability is dependant on the notification capability being supported.

6.3. Capability Identifier

The :interleave capability is identified by the following capability string:

urn:ietf:params:netconf:capability:interleave:1.0

6.4. New Operations

None.

6.5. Modifications to Existing Operations

When a <create-subscription> is sent while another subscription is active on that session, the following error will be 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.

7. Security Considerations

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

The access control framework and the choice of transport will have a major impact on the security of the solution.

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.

It is recommended that care be taken to secure execution:

- <create-subscription> invocation
- <get> on read-only data models
- <notification> content

Secure execution means ensuring that a secure transport is used as well as ensuring 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. A create <create-subscription> operation can be considered like a deferred <get>, and the content that different users can access may vary. This different access is reflected in the <notification> that different users are able to subscribe to.

One potential security issue is the transport of data from non-NETCONF streams, such as syslog and SNMP. This data may be more vulnerable (or less vulnerable) when being transported over NETCONF than when being transported using the protocol normally used for transporting it, depending on the security credentials of the two subsystems. The NETCONF server is responsible for applying access control to stream content.

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. If a user is not authorized to view all elements in the content of the notification, the notification is not sent to that user.

If a subscription is created with a <stopTime>, the NETCONF session will return to being a normal command-response NETCONF session when the replay is completed. It is the responsibility of the NETCONF

client to close this session when it is no longer of use.

8. IANA Considerations

-- Editor note to IANA/RFC-Editor: we request that you make these assignments, in which case it is to be documented as below

This document registers three URIs for the NETCONF XML namespace in the IETF XML registry [RFC3688].

Following the format in RFC 3688, IANA has made the following registration. Note that the capability urns as also compliant to [NETCONF] section 10.3.

Index	Capability Identifier
:notification	urn:ietf:params:netconf:capability: notification:1.0
:interleave	urn:ietf:params:netconf:capability: interleave:1.0

URI: urn:ietf:params:xml:ns:netmod:notification

URI: urn:ietf:params:xml:ns:netconf:notification:1.0

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

In addition, IANA registered the following XML Schema, the definition of which can be found in Section 4:
http://www.iana.org/assignments/xml-registry/schema/notification.xsd

9. Acknowledgements

Thanks to Gilbert Gagnon, Greg Wilbur and Kim Curran for providing their input into the early work on this document. In addition, the editors would like to acknowledge input at the Vancouver editing session from the following people: Orly Nicklass, James Balestriere, Yoshifumi Atarashi, Glenn Waters, Alexander Clemm, Dave Harrington, Dave Partain, Ray Atarashi and David Perkins and the following additional people from the Montreal editing session: Balazs Lengyel, Phil Shafer, Rob Enns, Andy Bierman, Dan Romascanu, Bert Wijnen, Simon Leinen, Juergen Schoenwaelder, Hideki Okita, Vincent Cridlig, Martin Bjorklund, Olivier Festor, Radu State, Brian Trammell, William Chow. We would also like to thank Li Yan for his numerous reviews as well as Suresh Krishnan for his gen-art review of the document.

10. Normative References

[NETCONF] Enns, R., "NETCONF Configuration Protocol", RFC 4741, December 2006.

[RFC2119] Bradner, s., "Key words for RFCs to Indicate Requirements Levels", RFC 2119, March 1997.

[RFC3339] Klyne, G., Ed. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, July 2002.

[RFC3688] Bradner, s., "The IETF XML Registry", RFC 3688, January 2004.

[XML] World Wide Web Consortium, "Extensible Markup Language (XML) 1.0", W3C XML, February 1998, <http://www.w3.org/TR/1998/REC-xml-19980210>.

[XML Schema] Thompson, H., Beech, D., Maloney, M., and N. Mendelsohn, "XML Schema Part 1: Structures Second Edition", W3C http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/structures.html, October 2004.

[XPATH] Clark, J. and S. DeRose, "XML Path Language (XPath) Version 1.0", W3C http://www.w3.org/TR/1999/REC-xpath-19991116, November 1999.

Appendix A. Change Log

-- Editor note to RFC-Editor: we request that you remove this section before publishing.

A.1. Version -08

1. Removed named profiles

2. Removed eventClass that was accidentally included in the definition of the replayComplete notification

3. Deleted data wrapper from notification

4. Changed replayLogStartTime to have a minOccurs of 0. It will only be there when replay is supported. Verify examples in section 3.2.5.1 are correct with respect to this element.

5. Error codes in section 2.1.1, fixed formatting issue

6. Moved replayComplete to not be under <netconf>

7. Section 2.1, fixed capitalization

8. In figure 4, the line was pushed out by 'system components', fixed this.

9. On page 8, replaced "If the startTime specified is earlier then the" with 'If the startTime specified is earlier than the"

10. Updated some name spaces and schemaLocations as per Andy's June 3rd email.

11. Added discussion of replayLogStartTime to draft in section 3.3.1 as follows "Whether or not a stream supports replay can be discovered by doing a <get> operation on the <streams> elements of the Notification Management Schema. This schema also provides the replayLogStartTime element to indicate the earliest available logged notification."

12. Removed most of the uses of the phrase 'Note that'. I kept two uses that prevent sentences from starting with either a lower case letter or an angle bracket.

13. In section 3.6 replaced "it will be filtered out" with "the notification will be filtered out"

14. In section 3.4, replaced "and the query" with "and to query"

15. Replaced 3 instances of "replay complete notification" with "replayComplete notification"

16. In section 3.3.2, replaced "normal NETCONF session" with "normal command-response NETCONF session"

17. In section 3.3.1, replaced "create an event subscription that will resend recently generated notification" with "create an event subscription that will resend recently generated notification, or is some cases send them for the first time to a particular NETCONF client."

18. In section 3.2.5.2, s/available event streams to/event streams available to/

19. In one spot, changed snmp to SNMP (the other gets deleted)

20. In section 3.2.5.1 s/where <name> element is/where the <name> element is/

21. In section 3.2.5.1, clarified that "value is unique" - within the scope of a NETCONF server.

22. In section 2.1.1, clarified that stopTime cannot preceded start time.

23. In section 2.1.1, in Start Time s/indicates/indicate/

24. In section 2.1.1, in Filter: s/This is mutually exclusive/The filter parameter is mutually exclusive/ ("this" could refer to the behaviour described in the previous sentence.)

25. In section 1.4, third bullet, replaced "syslog and SNMP are rather constrained in terms of message sizes)" with (ie, not too short)

26. In section 1.4, made all bullets start with capital letters.

27. Added definition of Filter to section 1.1

28. In section 1.1, improved the definition of subscription with "An agreement and method to receive event notifications over a NETCONF session."

29. In section 1.1, in the definition of operation, added a reference to [NETCONF].

30. Created a change log section

31. Fixed reference to IETF XML Registry in IANA Considerations section.

32. In section 3.3.3, deleted "This notification will only be sent if a 'stopTime' was specified when the replay subscription was created."

33. Added text to the security considerations section that says "If a subscription is created with a stopTime, the NETCONF session will return to being a normal command-response NETCONF session when the replay is completed. It is the responsibility of the NETCONF client to close off this session when it is no longer of use".

34. Update examples in section 5 to get rid of extra wrapper tag.

35. In section 2.1, replace "A NETCONF server is not required to process RPC requests on the session associated with the subscription until the notification subscription is done and may silently discard these requests." with "A NETCONF server is will not read RPC requests, by default, on the session associated with the subscription until the notification subscription is done.

36. Updated the notification definition and the replyComplete notification definition to use a substitution group.

A.2. Version -09

1. In section 5.1 "logical OR operation" -> "application of the logical OR operator"

2. In section 6 "ensure the secure operation of the following commands" -> "secure execution"

3. Removed a couple remaining references to named profiles.

4. Updated name datatype in eventStreams element.

5. Modified the cardinality of eventStreams to reflect that there will always be at least one event stream.

6. Fixed description of examples to remove reference to eventEntry, which is no longer part of the actual example.

7. In examples, for consistency changed some references to reportingElement to be reportingEntity

8. Fixed section 3.2, third paragraph to talk about filter elements instead of filters.

9. Merge section 3.3.2 and section 3.3.3. Delete the first paragraph in (old) section 3.3.3 since it both duplicates and contradicts text in section 3.3.2

10. In section 3.2.5.2.1, added clarification to first paragraph that "Either subtree or XPATH filtering can be used. "

11. Removed discussion of not allowing the return of stream names for which the user does not have permissions from the body of the document to the security considerations section.

12. Fixed typos and did wordsmithing in various parts of the document.

13. In section 2.1, explicitly stated that a subscription is bound to a single stream for the lifetime of the subscription.

14. removed single quotes around some instances of stopTime and startTime for consistency. When appropriate, put between angle brackets.

15. In section 2.1.1, changed "Error-info: <badElement>: startTime" to use bad-element.

16. In section 2.2.1, under the parameter tag, replaced "Contains notification-specific tagged content." with "Contains notification-specific tagged content, if any. "

17. Clarified some text in section 3.2, paragraph 3 around sending of filters from client and the filters later being applied to the notifications.

18. Fixed target namespace in section 4.

19. Added missing lang and version information to schema in section 3.4

20. Clarified that the examples in section 5 all used the same example event list.

21. Cleaned up security considerations section.

22. In section 3.4, clarified the definition of replayLogStart time to be the timestamp of the earliest available notification in the log used to support the replay function in the description tag for the object definition.

23. In section 3.3.2, clarified that the time an event was generated by the system means time an event was generated by the event source.

24. In section 3.5, deleted discussion about possibly defining subscriptions in XML Schema.

25. In section 3.6, deleted discussion about filter element execution order not mattering.

26. Fixed examples in section 5 to add <netconf> tag and to make other corrections

27. Added XML Schema definition for examples in section 5 and showed the event list with <notification> wrappers.

28. Added <notificationComplete> notification

29. Removed support of startTime and stopTime in the future.

30. Replaced replayLogStartTime with replayLogCreationTime and replayLogAgedTime.

A.3. Version -10

1. Changed the description of stopTime to allow stopTimes in the future.

2. Added interleave capability

3. Clarified create-subscription error messages.

4. Corrected targetNamespace in Netconf Notification XSD

5. Fixed typos and made minor edits.

A.4. Version -11

1. Fixed namespaces

2. In section 6.5, fixed error message Error-info

3. In section 6.1 clarify that if the interleave capability is supported, then the server must respond to requests.

A.5. Version -12

1. Add to section 1.3 the clarification "Note that a subscription cannot be modified once created."

2. In section 2.2.1, in the description of eventTime, added the following text: "This parameter is of type dateTime."

3. Fixed several typos.

4. Added the following text to the IANA considerations section: "-- Editor note to IANA/RFC-Editor: we request that you make these assignments, in which case it is top be documented as below" "

5. Replaced/Updated XML Schema reference to be " [XML Schema] Thompson, H., Beech, D., Maloney, M., Mendelsohn, N., "XML Schema Part 1: Structures Second Edition", W3C Recommendation, 28 October 2004 <http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/structures.html> "

6. Add instructions to RFC editor to remove change log before publication

7. Added IANA registration item for http://www.iana.org/assignments/xml-registry/schema/notification.xsd

8. Clarified in the IANA considerations section that the capability URIs were complaint to RFC4741 section 10.3

A.6. Version -13

1. In section 2.1.1, for both instances and in section 2.2.1, replaced "This parameter is of type dateTime." With "This parameter is of type dateTime and compliant to [RFC3339]."

2. In the normative reference section, added the following reference [RFC3339] Klyne, G., Ed. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, July 2002.

3. In section 2.1.1, for both instances and in section 2.2.1, added the following after the text \ indicating that the time fields are dateTime (this covers eventTime, startTime and stopTime). "Implementations must support time zones."

4. In Section 3.6.1 "parameter. A Filter only exist as a parameter to the subscription." s/exist/exists/

5. In Section 7: Replaced second last paragraph with "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. If a user is not authorized to view all elements in the content of the notification, the notification is not sent to that user."

6. In section 3.3.2, replaced "The NETCONF server will then accept <rpc> operations." With "The NETCONF server will then accept <rpc> operations even if the server did not previously accept such operations due to lack of interleave support."

7. In section 3.3.2, replaced "A <replayComplete> notification is sent to indicate that all of the replay notifications have been sent. If this subscription has a stop time, then this session becomes a normal NETCONF session again. When a <stopTime> has been specified, <notificationComplete> notification is the last notification sent on the subscription before it terminates and the NETCONF session returns to being a normal NETCONF session." With "A <replayComplete> notification is sent to indicate that all of the replay notifications have been sent. When a <stopTime> has been specified, <notificationComplete> notification is the last notification sent on the subscription before it terminates and the NETCONF session returns to being a normal NETCONF session. "

8. In section 3.3.2, replaced, "A <replayComplete> notification is sent to indicate that all of the replay notifications have been sent." With "A <replayComplete> notification is sent to indicate that all of the replay notifications have been sent and must not be sent for any other reason."

9. In section 3.3.1, replaced "A notification stream that supports replay is not expected to have an unlimited supply of saved notifications available to accommodate any replay request." With "A notification stream that supports replay is not expected to have an unlimited supply of saved notifications available to accommodate any replay request. Clients can query <replayLogCreationTime> and <replayLogAgedTime> to learn about the availability of notifications for replay."

10. In section 3.2, replaced "Figure 2 illustrates the notification flow and concepts identified in this document." With "Figure 2 illustrates the notification flow and concepts identified in this document. It does not mandate and/or preclude an

implementation."

11. In section 6.1, added the following text "This capability helps scalability by reducing the total number of NETCONF sessions required by a given operator or management application."

12. In section 3.6, deleted "When multiple filter elements are specified, they are applied collectively, so event notifications need to pass all specified filter elements in order to be sent to the subscriber. "

13. In section 7 (Security Considerations) after the bullets added the following: Secure execution means ensuring that a secure transport is used as well as ensuring that the user has sufficient authorization to perform the function they are requesting against the specific piece of NETCONF content involved. When a <get> is received against the content defined in this memo, clients should only be able to view the content for which they have sufficient privileges. A create <create-subscription> operation can be considered like a deferred <get>, and the content that different users can access may vary. This different access is reflected in the <notification> that different users are able to subscribe to.

14. Updated import statements to not used fully qualified URLs.

Authors' Addresses

Sharon Chisholm
Nortel
3500 Carling Ave
Nepean, Ontario K2H 8E9
Canada

Email: schishol@nortel.com

Hector Trevino
Cisco
Suite 400
9155 E. Nichols Ave
Englewood, CO 80112
USA

Email: htrevino@cisco.com

Full Copyright Statement

Copyright (C) The IETF Trust (2008).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).