Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 16, 2007

S. Chisholm
Nortel
H. Trevino
Cisco
May 15, 2007

NETCONF Event Notifications
draft-ietf-netconf-notification-07.txt

Status of this Memo

Copyright Notice

Abstract

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 document defines the capabilities and operations necessary to support this service.

Table of Contents

1. Introduction

[NETCONF] can be conceptually partitioned into four layers:

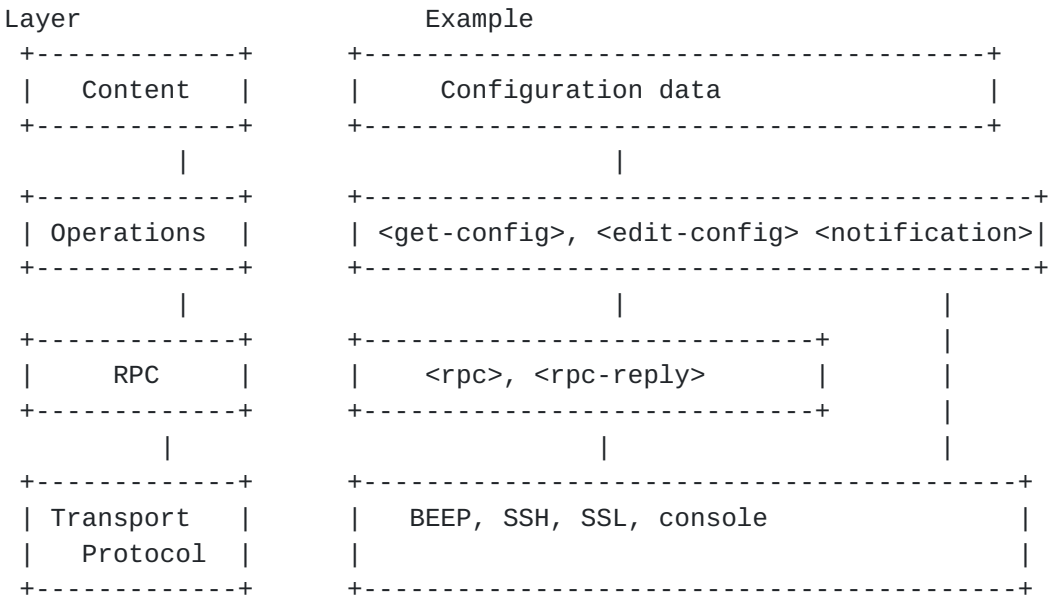

Figure 1

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: A concept related to the delivery of notifications (if 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. Specifically 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 it is available to NETCONF clients for subscription.

Named Profile: A predefined set of filtering criteria residing in the Network Element which may be applied to a notification session at subscription creation time.

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.

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.

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. A capability may be advertised to announce that a server is able to process RPCs while a notification stream is active on a session. The behaviour of such a capability is outside the scope of this document.

1.4. Requirements

The following requirements have been addressed by the solution:

- Initial release should ensure it supports notification in support of configuration operations
- Data content must not preclude the use of the same data model as used in configuration
- solution should support a reasonable message size limit (syslog and SNMP are rather constrained in terms of message sizes)
- solution should provide reliable delivery of notifications
- solution should provide a subscription mechanism (A NETCONF server does not send notifications before being asked to do so and the NETCONF client initiates the flow of notifications)
- solution should provide a filtering mechanism within the NETCONF server
- solution should send sufficient information in a notification so that it can be analyzed independent of the transport mechanism (data content fully describes a notification; protocol information is not needed to understand a notification)
- solution should support replay of locally logged notifications

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. 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 that indicates which stream of events is of interest. If not present, then events in the default NETCONF stream will be sent.

Filter:

An optional parameter that indicates which subset of all possible events are 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. This is mutually exclusive with the named profile parameter. See section 3.6 for more information on filters.

Named Profile:

An optional parameter that refers to a separately defined filter profile. If not present, no additional filtering will be applied. Note that changes to the profile after the subscription has been created will have no effect. This is mutually exclusive with the filter parameter. For more information on named profiles, see section 3.6.1.

Start Time:

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

Stop Time:

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'. It is valid to specify stop times that are later than the current time. This parameter is of type dateTime.

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 profile or 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: <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

2.1.1.1. Usage Example

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

Data:

Contains notification-specific tagged content. The content of the data tag 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 terminating the NETCONF session (<kill-session>) or the underlying transport session. If a stop time is provided when the subscription is created, then 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.

The diagram depicted in Figure 2 illustrates the notification flow and concepts identified in this document. 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.

When a NETCONF client subscribes to a given event stream, user-defined filters, if applicable, are applied to the event stream and matching event notifications are forwarded to the NETCONF server for

distribution to subscribed NETCONF clients. For more information on filters, 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.

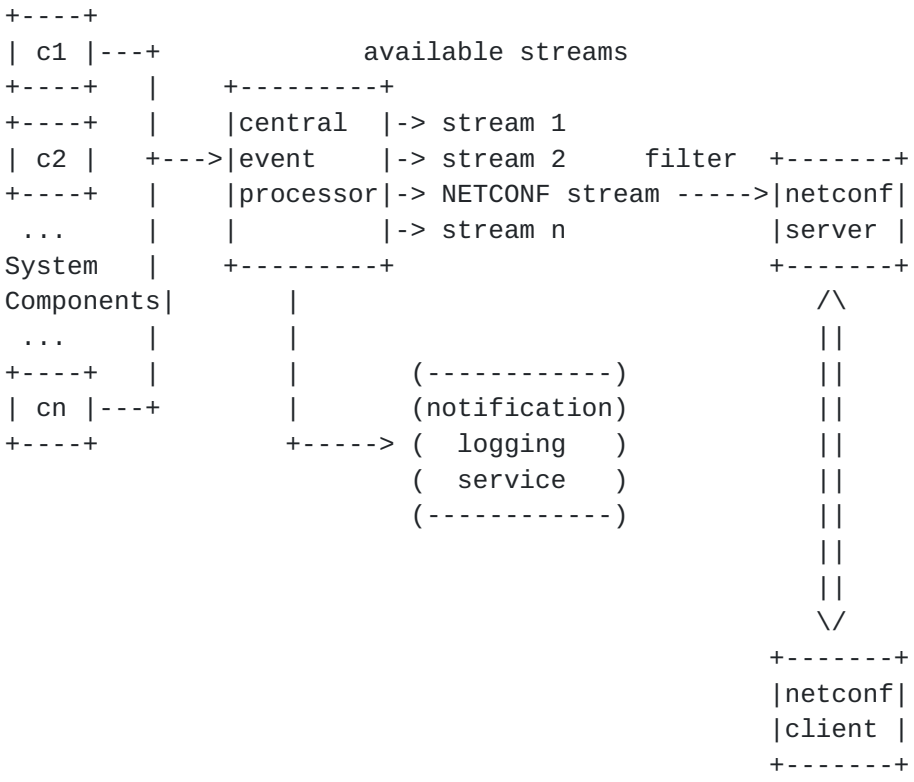

Figure 4

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) or user configurable (e.g., part of the device's configuration) or both. Device vendors may allow event stream configuration via 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 definition of the event notifications and their contents 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 notifications) specification of additional event stream sources (e.g., SNMP, syslog, etc.) 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> RPC request.

3.2.5.1. Name Retrieval using <get> operation

The list of available event streams is retrieved by requesting the <eventStreams> subtree via a <get> operation. Available event streams for the requesting session are returned in the reply containing the <name> and <description> elements, where <name> element is mandatory and its value is unique. The returned list must only include the names of those event streams for which the NETCONF session has sufficient privileges. The NETCONF session privileges are determined via access control mechanisms which are beyond the scope of this document. An empty reply is returned if there are no available event streams. The information is retrieved by requesting the <eventStreams> subtree via a <get> operation.

Example: Retrieving available event stream list using <get> operation:

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

  <get>
   <filter type="subtree">
     <eventStreams xmlns="urn:ietf:params:xml:ns:netmod:notification"/>
   </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>
      <eventStreams xmlns="urn:ietf:params:xml:ns:netmod:notification">
           <stream>
              <name>NETCONF</name>
              <description>Default netconf event stream
              </description>
              <replaySupport>true</replaySupport>
           </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>
           </stream>
         </eventStreams>
   </data>
 </rpc-reply>
```

3.2.5.2. Event Stream Subscription

A NETCONF client may request from the NETCONF server the list of available event streams 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 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.

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, then 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. These notifications are sent the same way as normal notifications.

A replay of notifications is specified by including an optional parameter to the subscription command that 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.

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 the current 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, nor where it resides within the device.

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 system.

A replay complete 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. In the case of a subscription without a stop time, after the replay complete 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.

3.3.3. Replay Complete Notification

The replay complete notification is the last notification sent over a replay subscription. It indicates that replay is complete. This notification will only be sent if a 'stopTime' was specified when the replay subscription was created. After this notification is received the subscription is terminated and the session becomes a normal NETCONF session.

The replayComplete can not be filtered out. It will always be sent on a relay subscription that specified a stop time.

3.4. Notification Management Schema

This Schema is used to learn about the event streams supported on the system and the query and modify named profiles. It also contains the definition of the replayComplete, which is sent to indicate that an event replay has sent all applicable notifications."

```
<?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:netconf:capability:notification:1.0"
```

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

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

<xs:complexType name="Netconf">
  <xs:sequence>
      <xs:element name="eventStreams" >
        <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="0" maxOccurs="unbounded">
             <xs:element name="stream">
                <xs:annotation>
                  <xs:documentation>
                    Stream name and description.
                  </xs:documentation>
                </xs:annotation>
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="name" type="xs:string"/>
                    <xs:element name="description"
                                        type="xs:string"/>
```

```
                    <xs:element name="replaySupport"
                                        type="xs:boolean"/>
                    <xs:element name="replayLogStartTime"
                                        type="xs:dateTime">
                      <xs:annotation>
                        <xs:documentation>
                           The start time of the log used to
                           support the replay function. If
                           replay is not supported, this will
                           return the current time.
                         </xs:documentation>
                       </xs:annotation>
                     </xs:element>
                   </xs:sequence>
                 </xs:complexType>
               </xs:element>
             </xs:sequence>
           </xs:complexType>
         </xs:element>


         <xs:element name="namedProfiles" minOccurs="0">
           <xs:complexType>
             <xs:sequence>
               <xs:element name="namedProfile"
                       type="manageEvent:NamedProfile" minOccurs="0"
                            maxOccurs="unbounded" />
             </xs:sequence>
           </xs:complexType>
         </xs:element>

       <xs:element name="replayComplete"
                   type="manageEvent:ReplayCompleteNotificationType"
                   minOccurs="0">
         <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:sequence>
   </xs:complexType>

   <xs:complexType name="ReplayCompleteNotificationType" >
       <xs:complexContent>
           <xs:extension base="ncEvent:NotificationType">
               <xs:sequence>
```

```
                    <xs:element name="eventClass"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>



    <xs:complexType name="NamedProfile">
        <xs:annotation>
            <xs:documentation>
                A named profile, which is a saved set of parameters
                that may be associated with zero or more
                active subscriptions.

                This object can be created, read, deleted and its
                individual components can be modified.
            </xs:documentation>
        </xs:annotation>
        <xs:sequence>

            <xs:element name="name">
                <xs:annotation>
                    <xs:documentation>
                        The name associated with the profile.
                        This object is readable and modifiable.
                    </xs:documentation>
                </xs:annotation>
            </xs:element>

            <xs:element name="stream"  minOccurs="0">
                <xs:annotation>
                    <xs:documentation xml:lang="en">
                        The event stream associated with this named
                        profile.

                        This object is readable and modifiable.
                    </xs:documentation>
                </xs:annotation>
            </xs:element>

            <xs:element name="filter"
                type="netconf:filterInlineType"  minOccurs="0">
                <xs:annotation>
                    <xs:documentation xml:lang="en">
                        The filters associated with this named
                        profile.
```

```
                        This object is readable and modifiable.
                    </xs:documentation>
                </xs:annotation>
            </xs:element>

            <xs:element name="lastModified" type="xs:dateTime">
                <xs:annotation>
                    <xs:documentation>
                        The timestamp of the last modification to this
                        named Profile. Note that modification of the
                        profile does not cause an immediate update
                        to all applicable subscription. Therefore,
                        this time should be compared with the last
                        modified time associated with the
                        subscription. If this time is earlier, then
                        the subscription is using the exact set of
                        parameters associated with this named profile.
                        If this time is later, then the subscription
                        is using an earlier version of this named
                        profile and the exact parameters may not
                        match. Note there is no guarantee that the
                        profile named in the subscription will be
                        found at all."

                        This object is read-only.
                    </xs:documentation>
                </xs:annotation>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

3.5. Subscriptions Data

While it may be possible to retrieve information about subscriptions via a get operation, subscriptions are not stored configuration. They are non-persistent state information and their lifetime is defined by their session.

Named profiles, if used, are considered configuration data.

3.6. Filter Mechanics

Note that when multiple filter elements are specified, they are applied collectively, so event notifications need to pass all specified filters in order to be sent to the subscriber. 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, it 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.

Note that the order that filter elements are applied does not matter since the resulting set of notifications is the intersection of the set of notifications that pass each filtering criteria.

3.6.1. Named Profiles

A named profile is a filter that is created ahead of time and applied at the time an event notification subscription is created. Note that changes to the profile after the subscription has been created will have no effect on the subscription. Since named profiles exist outside of the subscription, they persist after the subscription has been torn down.

3.6.2. Filtering

Inline filtering is explicitly stated when the event notification subscription is created. This is specified via the 'filter' parameter. Filters only exist as parameters to the subscription.

3.7. Message Flow

The following figure depicts message flow between a NETCONF client (C) and NETCONF server (S) in order create a subscription and begin the flow of notifications. It is possible that many rpc/rpc-reply sequences occur before the subscription is created or after a stopTime in a replay subscription, but this is not depicted in the figure.

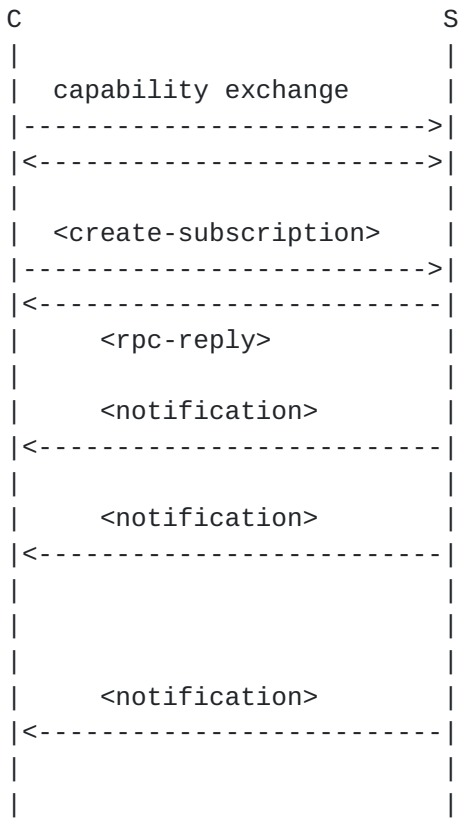

Figure 8

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:netconf:capability:notification:1.0"
        xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0"
        targetNamespace=
          "urn:ietf:params:netconf:capability: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="urn:ietf:params:xml:ns:netconf:base:1.0" />



<!-- ************** 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:choice>
            <xs:element name="filter"
                type="netconf:filterInlineType"
                minOccurs="0">
                <xs:annotation>
                    <xs:documentation>
                        An optional parameter that indicates
                        which subset of all possible events
                        are 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. This is
                        mutually exclusive with the named
                        profile parameter.
                    </xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element name="named-profile"
                type="namedProfileType" minOccurs="0">
                <xs:annotation>
                    <xs:documentation>
                        An optional parameter that points to
                        a separately defined filter profile.
                        If not present, no additional
                        filtering will be applied. Note that
                        changes to the profile after the
                        subscription has been created will
                        have no effect. This is mutually
                        exclusive with the filter parameter.
                    </xs:documentation>
                </xs:annotation>
            </xs:element>
        </xs:choice>
        <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="namedProfileType">
    <xs:annotation>
        <xs:documentation>
            The name of a saved profile containing filtering
            elements.
        </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string"/>
</xs:simpleType>

<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 or profile information. All 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.
            </xs:documentation>
        </xs:annotation>
    </xs:element>

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

     <!-- <Notification> operation -->
     <xs:complexType name="NotificationType">
        <xs:sequence>
          <xs:element name="data" type="netconf:dataInlineType" />
        </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.

5.1. Subtree Filtering

XML subtree filtering is not well suited for creating elaborate filter definitions given that it only supports equality comparisons and logical OR operations (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.

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

Sample event list

```
<events>
  <eventEntry>
    <eventClass>fault</eventClass>
    <reportingEntity>
      <card>Ethernet0</card>
    </reportingEntity>
    <severity>major</severity>
  </eventEntry>
  <eventEntry>
    <eventClass>fault</eventClass>
    <reportingEntity>
      <card>Ethernet2</card>
    </reportingEntity>
    <severity>critical</severity>
  </eventEntry>
  <eventEntry>
    <eventClass>fault</eventClass>
    <reportingEntity>
      <card>ATM1</card>
    </reportingEntity>
    <severity>minor</severity>
  </eventEntry>
  <eventEntry>
    <eventClass>state</eventClass>
    <reportingEntity>
      <card>Ethernet0</card>
    </reportingEntity>
    <operState>enabled</operState>
  </eventEntry>
</events>
```

The following example illustrates selecting events which have severities of critical, major, or minor (presumably fault events). The filtering criteria evaluation is as follows:

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

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

The following example illustrates selecting 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 message-id="101"
           xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
    <create-subscription>
       <filter netconf:type="subtree">
            xmlns="urn:ietf:params:netconf:capability:notification:1.0">
            <event xmlns="http://example.com/event/1.0">
                 <events>
                    <eventEntry>
                       <eventClass>fault</eventClass>
                    </eventEntry>
                    <eventEntry>
                       <eventClass>state</eventClass>
                    </eventEntry>
                    <eventEntry>
                       <eventClass>config</eventClass>
                    </eventEntry>
                    <eventEntry>
                      <eventClass>fault</eventClass>
                        <reportingElement>
                           <card>Ethernet0</card>
                        </reportingElement>
                    </eventEntry>
                 </events>
            </event>
       </filter>
    </create-subscription>
 </netconf:rpc>
```

5.2. XPATH filters

The following [XPATH] example illustrates selecting 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 message-id="101"
             xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
      <create-subscription>
         <filter netconf:type="xpath"
           select="//eventEntry[(eventClass='fault' and
                 (severity='minor' or severity='major'
                  or severity='critical'))]"/>
      </create-subscription>
   </netconf:rpc>
```

The following example illustrates selecting state and config EventClasses or fault events that have severities of critical, major, or minor or come from card Ethernet0. The filtering criteria evaluation is as follows:

((state | config) & ((fault & severity=critical) | (fault & severity=major) | (fault & severity = minor) | (fault & card=Ethernet0)))

```
<netconf:rpc message-id="101"
               xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
      <create-subscription>
         <filter netconf:type="xpath"
                     select="//eventEntry[(eventClass='fault' and
                     severity='minor') or
                     (eventClass='fault' and severity='major') or
                     (eventClass='fault' and severity='critical') or
                     (eventClass='fault' and card='Ethernet0') or
                      eventClass='state' or eventClass='config']"/>
     </create-subscription>
   </netconf:rpc>
```

6. Security Considerations

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

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

Note that the <notification> elements are never sent before the transport layer and the netconf layer (capabilities exchange) have been established, and the manager has been identified and authenticated.

It is recommended that care be taken to ensure the secure operation of the following commands:

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

One issue related to the notifications draft is the transport of data from non-netconf streams, such as syslog and SNMP. Note that this data may be more vulnerable (or is not more 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.

If a user does not have permission to view content via other NETCONF operations it does not have permission to access that content via Notifications. If a user is not permitted to view one element in the content of the notification, the notification is not sent to that user.

7. IANA Considerations

This document registers two URIs for the NETCONF XML namespace in the IETF XML registry [7].

Following the format in RFC 3688, IANA has made the following registration.

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

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

Registrant Contact: The IESG.

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

8. 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 Dave 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.

9. Normative References

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

[RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", RFC 2026, BCP 9, October 1996.

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

[RFC2223] Postel, J. and J. Reynolds, "Instructions to RFC Authors", RFC 2223, October 1997.

[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] Fallside, D. and P. Walmsley, "XML Schema Part 0: Primer Second Edition", W3C XML Schema, 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.

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

Intellectual Property

Acknowledgment

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