Network Working Group
Internet-Draft
Expires: March 18, 2007

S. Chisholm
Nortel
H. Trevino
Cisco

September 14, 2006

NETCONF Event Notifications
draft-ietf-netconf-notification-03.txt

Status of this Memo

Copyright Notice

Abstract

This memo defines a framework for sending asynchronous messages, or event notifications in NETCONF. It defines both the operations necessary to support this concept, and also discusses implications for the mapping to transport protocols.

Table of Contents

1. Introduction

NETCONF [NETCONF-PROTO] can be conceptually partitioned into four layers:

```
   Layer                      Example
    +-------------+      +----------------------------------------+
    |   Content   |      |     Configuration data                 |
    +-------------+      +----------------------------------------+
              |                           |
    +-------------+      +-------------------------------------------+
    | Operations  |      | <get-config>, <edit-config> <notification>|
    +-------------+      +-------------------------------------------+
              |                           |                    |
    +-------------+      +-----------------------------+       |
    |     RPC     |      |    <rpc>, <rpc-reply>       |       |
    +-------------+      +-----------------------------+       |
             |                            |                    |
    +-------------+      +------------------------------------------+
    | Transport   |      |   BEEP, SSH, SSL, console                |
    |   Protocol  |      |                                          |
    +-------------+      +------------------------------------------+
```

This document defines a framework for sending asynchronous messages, or event notifications in NETCONF. It defines both the operations necessary to support this concept, and also discusses implications for the mapping to transport protocols.

Figure 1

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 RFC 2119 [3].

Element: An XML Element[XML].

Managed Entity: A node, which supports NETCONF[NETCONF-PROTO] and has access to management instrumentation. This is also known as the NETCONF server.

Managed Object: A collection of one of more Elements that define an abstract thing of interest.

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.

1.2 Event Notifications in NETCONF

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 out to interested parties to notify them that this event has occurred.

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 some event, outside the scope of this specification, causes the subscription to terminate. 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.

An agent is not required to process RPC requests until the notification stream is done. A capability may be defined to announce that a server is able to process RPCs while a notification stream is active on a session.

1.3 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.

1.4 Requirements

The requirements for this solution are as follows:

o Initial release should ensure it supports notification in support of configuration operations

o 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 support agent initiated connections
- solution should provide a subscription mechanism (An agent does not send notifications before asked to do so and the manager initiates the flow of notifications)
- solution should provide a filtering mechanism
- 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 not bind subscriptions to a connection
- channels for configuration change notifications should share fate with a session that includes a configuration channel
- 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

<create-subscription>

Description:

This operation initiates an event notification subscription which will send asynchronous event notifications to the initiator of the command until the NETCONF session terminates or some event, outside the scope of this specification, causes the subscription to terminate.

Parameters:

Stream:

An optional parameter that indicates which stream(s) of events are 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.

Named Profile

An optional parameter that points to a separately defined filter profile. The contents of the profile are specified in the provided XML Schema. 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.

Positive Response:

If the NETCONF server can satisfy the request, the server sends an <rpc-reply> element containing a <data> element containing the subscription ID.

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.

2.2 Sending Event Notifications

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

2.2.1 Event Notification

<notification>

Description:

An event notification is sent to the initiator of an <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 XML document.

Parameters:

Subscription Id:

A unique identifier for this event subscription

Data:

Contains notification-specific tagged content.

Positive Response:

No response.

Negative Response:

No response.

2.3 Terminating the Subscription

Closing of the event notification subscription is done by terminating the Netconf session (<kill-session>)or via some action outside the scope of this specification.

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.

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

For 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:xml:ns:netconf:capability:startup:1.0
    </capability>
    <capability>
      urn:ietf:params:xml:ns:netconf:notification:1.0
    </capability>
  </capabilities>
  <session-id>4</session-id>
</hello>
```

3.2 Event Streams

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

System components 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. 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.

```
   +----+
   | c1 |---+             available streams
   +----+   |    +---------+
   +----+   |    |central  |-> stream 1
   | c2 |   +--->|event    |-> stream 2    filter +-------+
   +----+   |    |processor|-> netconf stream --->|netconf|
    ...     |    |         |-> stream n           |server | see
   System   |    +---------+                      +-------+ below
   Components|        |                  //
    ...     |        |                //
   +----+   |        |       (------------)
   | cn |---+        |       (notification)
   +----+            +-----> (  logging   )
                             (  service   )
                             (------------)



                +-------+     +-------+
                |netconf|<--->|netconf|
             -> |server |     |client |
                +-------+     +-------+
```

3.2.1 Event Stream Definition

Event streams are pre-defined 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 notification 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> or <get-config> RPC request.

3.2.5.1 Name Retrieval using get, get-config RPC

The list of available event streams is retrieved by requesting the <eventStreams> subtree via a <get> or <get-config> operation. Available event streams for the requesting session are returned in the reply containing <name> and <description> elements, where <name> element is mandatory and its value is unique [Editor's Note: should we then define it as a key?]. The returned list must only include the names of those event streams for which the NETCONF sessions 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.

   Retrieving available event stream list using <get-config> operation:

```
            <get-config>
            <source>
              <running/>
            </source>
            <filter type="subtree">
              <top xmlns="http://example.com/schema/1.2/config">
                 <sessionEventStream>
                <eventStreams/>
                </sessionEventStream>
              </top>
            </filter>
          </get-config>
        </rpc>


        <rpc-reply message-id="101"
             xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
          <data>
            <top xmlns="http://example.com/schema/1.2/config">
              <sessionEventStream>
              <eventStreams>
                <stream>
                  <name>NETCONF</name>
                  <description>Default netconf event stream
                    </description>
                </stream>
                <stream>
                  <name>snmp</name>
                  <description>SNMP notifications</description>
                </stream>
                <stream>
                  <name>syslog-critical</name>
                  <description>Critical and higher severity
                    </description>
                </stream>
               </sessionEventStreams>
              </eventStreams>
            </top>
          </data>
        </rpc-reply>
```

Retrieving available event stream list using <get> operation:

```
              <get>
                <filter type="subtree">
                  <top xmlns="http://example.com/schema/1.2/config">
                     <sessionEventStreams>
                    <eventStreams/>
                     </sessionEventStreams>
                  </top>
                </filter>
              </get>
              </rpc>


               <rpc-reply message-id="101"
                 xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
                 <data>
                   <top xmlns="http://example.com/schema/1.2/config">
                     <sessionEventStreams>
                    <eventStreams>
                      <stream>
                         <name>NETCONF</name>
                        <description>Default netconf event stream
                        </description>
                       </stream>
                       <stream>
                        <name>snmp</name>
                        <description>SNMP notifications</description>
                       </stream>
                       <stream>
                         <name>syslog-critical</name>
                         <description>Critical and higher severity
                         </description>
                       </stream>
                     </eventStreams>
                     </sessionEventStreams>
                   </top>
                 </data>
               </rpc-reply>
```

3.2.5.2 Device Supported Event Streams (System)

The list of all event streams configured on a device may be retrieved over a NETCONF session with sufficient privileges (e.g. administrator). The information is retrieved by requesting the

<systemEventStreams> subtree via a <get> or <get-config>
operation.

```
                 <get-config>
                   <source>
                     <running/>
                   </source>
                   <filter type="subtree">
                 <top xmlns="http://example.com/schema/1.2/config">
                       <systemEventStreams/>
                     </top>
                   </filter>
                 </get-config>
                 </rpc>



         <rpc-reply message-id="101"
                   xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
           <data>
               <top xmlns="http://example.com/schema/1.2/config">
                  <systemEventStreams>
                     <stream>
                        <name>NETCONF</name>
                         <description>Default netconf event stream
                         </description>
                         </stream>
                         <stream>
                         <name>snmp</name>
                         <description>SNMP notifications
                         </description>
                         </stream>
                         <stream>
                         <name>syslog-critical</name>
                        <description>Critical and higher severity
                        </description>
                      </stream>
                   </systemEventStreams>
                 </top>
             </data>
          </rpc-reply>
```

3.2.5.3 **Stream Retrieval Schema**

```
   <?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"
          elementFormDefault="qualified"
                    attributeFormDefault="unqualified">
           <xs:annotation>
                   <xs:documentation xml:lang="en">
                        Schema for event streams
            </xs:documentation>
                   <xs:appinfo>
                           <nm:identity
               xmlns:nm="urn:ietf:params:xml:ns:netmod:base:1.0">
                                   <nm:Name>
                    NetconfNotificationSchema
           </nm:Name>
                                   <nm:LastUpdated>
                    2006-09-06T09:30:47-05:00
           </nm:LastUpdated>
                                   <nm:Organization>IETF
           </nm:Organization>
                                   <nm:Description>
                     A schema that can be used to learn about current
                     NetConf Event subscriptions and creating named
                     profiles
                  </nm:Description>
                           </nm:identity>
                   </xs:appinfo>
           </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="./draft-ietf-netconf-prot-12.xsd"/>


           <xs:element name="sessionEventStreams">
                   <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 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:sequence>
                                           </xs:complexType>
                                   </xs:element>
                           </xs:sequence>
                   </xs:complexType>
           </xs:element>
   </xs:schema>
```

3.2.6 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.

3.2.6.1 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.

3.2.6.2 Subscription to Multiple Event Streams

Multiple event streams may be configured on a device and a NETCONF client may subscribe to one or more of the available event streams. A NETCONF client subscribing to multiple event streams must do so by either establishing a new NETCONF session or opening a new channel on an existing NETCONF session.

3.3 Subscriptions and Datastores

Subscriptions are like NETCONF sessions in that they don't exist in

NETCONF datastores. The exception to this is the named profiles feature.

3.4 Querying Subscription Properties

The following Schema can be used to retrieve information about active event notification subscriptions

```
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:nsub="urn:ietf:params:xml:ns:netconf:subscription:1.0"
  targetNamespace="urn:ietf:params:xml:ns:netconf:subscription:1.0"
  xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0"
  xmlns:ncEvent="urn:ietf:params:xml:ns:netconf:notification:1.0"
  xmlns:nm="urn:ietf:params:xml:ns:netconf:appInfo:1.0"
  elementFormDefault="qualified" attributeFormDefault="unqualified"
  xml:lang="en">
  <xs:annotation>
       <xs:documentation xml:lang="en">
           Schema for reporting on Event Subscriptions
       </xs:documentation>
       <xs:appinfo>
           <nm:identity
               xmlns:nm="urn:ietf:params:xml:ns:netmod:base:1.0">
               <nm:Name>NetconfNotificationSchema</nm:Name>
               <nm:LastUpdated>2006-09-13T09:30:47-05:00
               </nm:LastUpdated>
               <nm:Organization>IETF</nm:Organization>
               <nm:Description>
                   A schema that can be used to learn about current
                   NetConf Event subscriptions and creating named
                   profiles
               </nm:Description>
           </nm:identity>
       </xs:appinfo>
   </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:notification:1.0"
      schemaLocation=
        "urn:ietf:params:xml:ns:netconf:notification:1.0"/>
   <xs:import namespace="urn:ietf:params:xml:ns:netconf:base:1.0"
      schemaLocation="urn:ietf:params:xml:ns:netconf:base:1.0"/>

   <!-- Associations -->
```

```
        <xs:element name="associatedNamedProfile" type="xs:string"/>

        <xs:element name="relationships">
        <xs:keyref name="subscriptionToNamedProfile"
             refer="nsub:namedProfileKey">
            <xs:selector xpath=".//netconfSubscription"/>
            <xs:field xpath="nsub:associatedNamedProfile"/>
        </xs:keyref>

        <!-- Keys -->

        <xs:key name="namedProfileKey">
            <xs:selector xpath=".//namedProfile"/>
            <xs:field xpath="name"/>
        </xs:key>
        </xs:element>

        <xs:element name="netconfSubscription">
            <xs:annotation>
                <xs:appinfo>
                    <nm:minAccess><read/></nm:minAccess>
                    <nm:maxAccess><read/></nm:maxAccess>
                </xs:appinfo>
            </xs:annotation>
            <xs:complexType>
                <xs:sequence >

            <xs:element name="session-id"
                    type="netconf:SessionId" >
               <xs:annotation>
                   <xs:documentation xml:lang="en">
                      The session id associated with this subscription.
                  </xs:documentation>
               </xs:annotation>
             </xs:element>

             <xs:element name="subscriptionID"
                        type="ncEvent:SubscriptionID" >
                <xs:annotation>
                   <xs:documentation xml:lang="en">
                     The subscription id associated with this
                     subscription.
                   </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 subscription.
                   </xs:documentation>
               </xs:annotation>
           </xs:element>

          <xs:element ref="nsub:associatedNamedProfile" minOccurs="0">
             <xs:annotation>
                <xs:documentation xml:lang="en">
                  The named profile associated with this subscription.
                  Note that the  contents of the named profile may
                  have changed since it was last applied.
                  </xs:documentation>
              </xs:annotation>
          </xs:element>

          <xs:element name="lastModified"
                            type="xs:dateTime" >
               <xs:annotation>
                  <xs:documentation xml:lang="en">
                     The last time this subscription was modified. If it
                     has not been modified since creation, this is the
                     time of subscription creation.
                    </xs:documentation>
                </xs:annotation>
            </xs:element>

            <xs:element name="messagesSent"
                            type="xs:integer" minOccurs="0">
                <xs:annotation>
                   <xs:documentation xml:lang="en">
                       A count of event notifications sent along
                       this connection since the subscription was
                       created.
                    </xs:documentation>
                    </xs:annotation>
               </xs:element>

               <xs:element name="key">
                   <xs:key name="uniqueSubscription">
                        <xs:selector xpath=".//subscription"/>
                         <xs:field xpath="session-id"/>
                          <xs:field xpath="subscriptionID"/>
                     </xs:key>
                </xs:element>
              </xs:sequence>
```

```
            </xs:complexType>
        </xs:element>

        <xs:element name="netconfSubscriptions">
            <xs:complexType>
                <xs:sequence>
                    <xs:element ref="nsub:netconfSubscription"
                        minOccurs="0"
                        maxOccurs="unbounded" />
                </xs:sequence>
            </xs:complexType>
        </xs:element>

        <xs:element name="namedProfile">
            <xs:annotation>
                <xs:appinfo>
                    <nm:minAccess><read/></nm:minAccess>
                    <nm:maxAccess><read/> <write/> <create/> <delete/>
                    </nm:maxAccess>
                </xs:appinfo>
            </xs:annotation>
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="name"/>
                    <xs:element name="filter"
                        type="netconf:filterInlineType"  minOccurs="0">
                        <xs:annotation>
                            <xs:documentation xml:lang="en">
                                The filters associated with this named
                                profile.
                            </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.
                      </xs:documentation>
                      <xs:appinfo>
                         <nm:minAccess><read/></nm:minAccess>
                         <nm:maxAccess><read/> </nm:maxAccess>
                     </xs:appinfo>
                 </xs:annotation>
               </xs:element>
                </xs:sequence>
            </xs:complexType>
        </xs:element>

        <xs:element name="namedProfiles">
            <xs:complexType>
                <xs:sequence>
                    <xs:element ref="nsub:namedProfile" minOccurs="0"
                        maxOccurs="unbounded" />
                </xs:sequence>
            </xs:complexType>
        </xs:element>


    </xs:schema>
```

3.5 One-way Notification Messages

In order to support the concept that each individual event notification is a well-defined XML-document that can be processed without waiting for all events to come in, it makes sense to define events, not as an endless reply to a subscription command, but as independent messages that originate from the NETCONF server. In order to support this model, this memo introduces the concept of notifications, which are one-way messages.

A one-way message is similar to the two-way RPC message, except that no response is expected to the command. In the case of event notification, this message will originate from the NETCONF server, and not the NETCONF client.

3.6 Filter Dependencies

Note that when multiple filters are specified (in-line Filter, Named Profiles), they are applied collectively, so event notifications need to pass all specified filters in order to be sent to the subscriber. If a filter 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 filters 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

Just-in-time 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.

```
                         C                           S
                         |                           |
                         |  capability exchange      |
                         |-------------------------->|
                         |<------------------------->|
                         |                           |
                         |  <create-subscription>    |
                         |-------------------------->|
                         |<--------------------------|
                         |     <rpc-reply>           |
                         |                           |
                         |     <notification>        |
                         |<--------------------------|
                         |                           |
                         |     <notification>        |
                         |<--------------------------|
                         |                           |
                         |                           |
                         |                           |
                         |     <notification>        |
                         |<--------------------------|
                         |                           |
                         |                           |
```

4. XML Schema for 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="urn:ietf:params:xml:ns:netconf:base:1.0" />


   <!-- ************** Type definitions ***********************-->

        <xs:simpleType name="SubscriptionID">
        <xs:annotation>
          <xs:documentation>
          The unique identifier for this particular subscription within
          the session.
          </xs:documentation>
          </xs:annotation>
           <xs:restriction base="xs:string"/>
           </xs:simpleType>

           <xs:simpleType name="SequenceNumber">
        <xs:annotation>
          <xs:documentation>
          A monotonically  increasing integer. Starts at 0.
          Always increases by just one. Roll back to 0 after maximum
          value is reached.
          </xs:documentation>
```

```
          </xs:annotation>
           <xs:restriction base="xs:integer"/>
           </xs:simpleType>




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


        <!--
          <create-subscription> operation
          -->
        <xs:complexType name="createSubscriptionType">
          <xs:complexContent>
            <xs:extension base="netconf:rpcOperationType">
              <xs:sequence>
                 </xs:element>
                <xs:element name="filter"
                     type="netconf:filterInlineType" minOccurs="0"/>
                <xs:element name="named-profile"
                            type="xs:string" minOccurs="0"/>
                <xs:element name="startTime" type="xs:dateTime"
                        minOccurs="0" />
              </xs:sequence>
            </xs:extension>
          </xs:complexContent>
        </xs:complexType>
        <xs:element name="create-subscription"
                    type="createSubscriptionType"
                    substitutionGroup="netconf:rpcOperation"/>




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

          <!--
          <Event> operation
          -->
      <xs:complexType name="NotificationType">
              <xs:sequence>
          <xs:element name="subscriptionId" type="SubscriptionID" />
              </xs:sequence>
        </xs:complexType>
        <xs:element name="notification" type="NotificationType"/>

      </xs:schema>
```

5. Mapping to Transport Protocols

Currently, the NETCONF family of specification allows for running NETCONF over a number of transport protocols, some of which support multiple configurations. Some of these options will be better suited for supporting event notifications then others.

5.1 SSH

Session establishment and two-way messages are based on the NETCONF over SSH transport mapping [NETCONF-SSH]

One-way event messages are supported as follows: Once the session has been established and capabilities have been exchanged, the server may send complete XML documents to the NETCONF client containing notification elements. No response is expected from the NETCONF client.

As the examples in [NETCONF-SSH] illustrate, a special character sequence, MUST be sent by both the client and the server after each XML document in the NETCONF exchange. This character sequence cannot legally appear in an XML document, so it can be unambiguously used to identify the end of the current document in the event notification of an XML syntax or parsing error, allowing resynchronization of the NETCONF exchange.

The NETCONF over SSH session to receive an event notification might look like the following. Note the event notification contents (delimited by <data> </data> tags) are not defined in this document and are provided herein simply for illustration purposes:

```
       <?xml version="1.0" encoding="UTF-8"?>
          <notification
                xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
            <subscription-id>123456</subscription-id>
            <data>
            <eventClasses><configuration/><audit/></eventClasses>
            <sequenceNumber>2</sequenceNumber>
            <dateAndTime>2000-01-12T12:13:14Z</dateAndTime>
                 <user>Fred Flinstone</user>
                 <operation>
                  <edit-config>
                    <target>
                     <running/>
                    </target>
                    <config>
                      <top xmlns="http://example.com/schema/1.2/config">
                         <interface>
                           <name>Ethernet0/0</name>
                           <mtu>1500</mtu>
                        </interface>
                      </top>
                   </config>
                 </edit-config>
               </operation>
             </data>
          </notification>
        ]]>
    ]]>
```

5.2 BEEP

Session establishment and two-way messages are based on the NETCONF over BEEP transport mapping NETCONF-BEEP

5.2.1 One-way Notification Messages in Beep

One-way notification messages can be supported either by mapping to the existing one-to-many BEEP construct or by creating a new one-to-none construct.

This area is for future study.

5.2.1.1 One-way messages via the One-to-many Construct

Messages in one-to-many exchanges: "rpc", "notification", "rpc-reply"

Messages in positive replies: "rpc-reply", "rpc-one-way"

5.2.1.2 One-way notification messages via the One-to-none Construct

Note that this construct would need to be added to an extension or update to 'The Blocks Extensible Exchange Protocol Core' RFC 3080.

MSG/NoANS: the client sends a "MSG" message, the server, sends no reply.

In one-to-none exchanges, no reply to the "MSG" message is expected.

5.3 SOAP

Session management and message exchange are based on the NETCONF over SOAP transport mapping NETCONF-SOAP

Note that the use of "persistent connections" "chunked transfer-coding" when using HTTP becomes even more important in the supporting of event notifications

5.3.1 A NETCONF over Soap over HTTP Example

```
C: POST /netconf HTTP/1.1
C: Host: netconfdevice
C: Content-Type: text/xml; charset=utf-8
C: Accept: application/soap+xml, text/*
C: Cache-Control: no-cache
C: Pragma: no-cache
C: Content-Length: 465
C:
C: <?xml version="1.0" encoding="UTF-8"?>
C: <soapenv:Envelope
C:   xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
C:   <soapenv:Body>
C:     <rpc message-id="101"
C:        xmlns=
        "xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
C:       <create-subscription>
C:       </create-subscription>
C:     </rpc>
C:   </soapenv:Body>
C: </soapenv:Envelope>
```

The response:

```
S: HTTP/1.1 200 OK
S: Content-Type: application/soap+xml; charset=utf-8
S: Content-Length: 917
S:
```

```
      S: <?xml version="1.0" encoding="UTF-8"?>
      S: <soapenv:Envelope
      S:   xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
      S:   <soapenv:Body>
      S:     <rpc-reply message-id="101"
      S:        xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
      S:       <data>
      S:         <top xmlns=
                      "http://example.com/schema/1.2/notification">
      S:           <subscriptionId>123456</subscriptionId>
      S:         </top>
      S:       </data>
      S:     </rpc-reply>
      S:   </soapenv:Body>
      S: </soapenv:Envelope>
```

And then some time later

```
      S: HTTP/1.1 200 OK
      S: Content-Type: application/soap+xml; charset=utf-8
      S: Content-Length: 917
      S:
      S: <?xml version="1.0" encoding="UTF-8"?>
      S: <soapenv:Envelope
      S:   xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
      S:   <soapenv:Body>
      S:     <notification
                xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
      S:      <subscriptionID>123456</subscriptionID>
      S:       <data>
      S:      <eventClasses><configuration/><audit/></eventClasses>
      S:      <sequenceNumber>2</sequenceNumber>
      S:           <dateAndTime>2000-01-12T12:13:14Z</dateAndTime>
      S:           <user>Fred Flinstone</user>
      S:              <operation>
      S:               <edit-config>
      S:              <target>
      S:               <running/>
      S:              </target>
      S:             <config>
      S:              <top xmlns="http://example.com/schema/1.2/config">
      S:                   <interface>
      S:                     <name>Ethernet0/0</name>
      S:                     <mtu>1500</mtu>
      S:                  </interface>
      S:               </top>
      S:            </config>
      S:           </edit-config>
```

```
      S:         </operation>
      S:       </data>
      S:    </notification>
      S:   </soapenv:Body>
      S: </soapenv:Envelope>
```

6. Filtering examples

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

6.1 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 (only for example purposes). The examples herein assume that the event notification schema definition has an <eventClasses> element identifying the event category (e.g. fault, state, config, etc.) and events have a <severity> element

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))

```
     <rpc message-id="101"
          xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
       <create-subscription>
         <netconf:filter type="subtree">
           <neb
            xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
             <event>
                 <severity>critical</severity>
             </event>
             <event>
                 <severity>major</severity>
             </event>
             <event>
                 <severity>minor</severity>
             </event>
           </neb>
         </netconf:filter>
       </create-subscription>
     </rpc>
```

The following example illustrates selecting fault, state, config EventClasses or events which are related to card Ethernet0. The filtering criteria evaluation is as follows:

(fault | state | config | card=Ethernet0)

```
     <rpc message-id="101"
          xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
       <create-subscription>
         <netconf:filter type="subtree">
           <neb
             xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
             <event>
                 <eventClasses>fault</eventClasses>
             </event>
             <event>
                 <eventClasses>state</eventClasses>
             </event>
             <event>
                 <eventClasses>config</eventClasses>
             </event>
             <event>
                 <card>Ethernet0</card>
             </event>
           </neb>
         </netconf:filter>
       </create-subscription>
     </rpc>
```

6.2 XPATH filters

The following 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)))

```
        <rpc message-id="101"
             xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
          <create-subscription>
            <netconf:filter type="xpath">
              (/event[eventClasses/fault] and
              (/event[severity="critical"] or
               /event[severity="major"] or /event[severity="minor"]))
            </netconf:filter>
          </create-subscription>
        </rpc>
```

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

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

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

7. Notification Replay Capability

7.1 Overview

Replay is the ability to create an event subscription that will resend recently sent 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 of the replay is implicitly defined to be the time the replay request was initiated.

An implementation that supports replay is not expected to have an unlimited supply of saved notifications available to accommodate any replay request. If a client requests a replay of notifications that predate the oldest notification available, then the NETCONF server must return an warning message in the RPC reply and start replaying the notifications it does have available, within the other constraints, such as filtering, that the client has provided.

The actual number of stored notifications available for retrieval at any given time is an agent implementation specific matter. Control parameters for this aspect of the feature are outside the scope of the current work.

A given subscription is either a replay subscription or a normal subscription, which sends event notifications as they happen. A replay subscription terminates once the it has completed replaying past events.

7.2 Dependencies

This capability is dependent on the notification capability being supported. It also requires that the device support some form of notification logging, although it puts no restrictions on the size or form of the log.

7.3 Capability Identifier

The Event Notification Replay capability is identified by following capability string:

http://ietf.org/netconf/notificationReplay/1.0

7.4 New Operations

None

7.5 Modifications to Existing Operations

7.5.1 create-subscription

This capability adds an optional parameter to the <create-subscription> command called 'startTime'. This identifies the earliest date and time of interest for event notifications being replayed. Events generated before this time are not matched.

7.5.2 Interactions with Other Capabilities

[Edtitor's Note: If this capability does not interact with other capabilities, this section may be omitted.]

8. Security Considerations

To be determined once specific aspects of this solution are better understood. In particular, the access control framework and the choice of transport will have a major impact on the security of the solution

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 Dave Perkins and the following additional people from the Montreal editing session: Balazs Lengyel, Phil Shafer, Rob Ennes, Andy Bierman, Dan Romascanu, Bert Wijnen, Simon Leinen, Juergen Schoenwaelder, Hideki Okita, Vincent Cridlig, Martin Bjorklund, Olivier Festor, Radu State, Brian Trammell, William Chow

10. References

[NETCONF] Enns, R., "NETCONF Configuration Protocol", ID draft-ietf-netconf-prot-12, February 2006.

[NETCONF BEEP] Lear, E. and K. Crozier, "Using the NETCONF Protocol over Blocks Extensible Exchange Protocol (BEEP)", ID draft-ietf-netconf-beep-10, March 2006.

[NETCONF Datamodel] Chisholm, S. and S. Adwankar, "Framework for NETCONF Content", ID draft-chisholm-netconf-model-05.txt, April 2006.

[NETCONF SOAP] Goddard, T., "Using the Network Configuration Protocol (NETCONF) Over the Simple Object Access Protocol (SOAP)", ID draft-ietf-netconf-soap-08, March 2006.

[NETCONF SSH] Wasserman, M. and T. Goddard, "Using the NETCONF Configuration Protocol over Secure Shell (SSH)", ID draft-ietf-netconf-ssh-06.txt, March 2006.

[URI] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998.

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

[refs.RFC2026]

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

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

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

[refs.RFC3080]
Rose, M., "The Blocks Extensible Exchange Protocol Core", RFC 3080, March 2001.

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

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.