NETCONF
Internet-Draft
Intended status: Standards Track
Expires: February 23, 2019

E. Voit
Cisco Systems
H. Birkholz
Fraunhofer SIT
A. Bierman
YumaWorks
A. Clemm
Huawei
T. Jenkins
Cisco Systems

August 22, 2018

Notification Message Headers and Bundles
draft-ietf-netconf-notification-messages-04

Abstract

This document defines a new notification message format, using yang-data. Included are:

- a new notification mechanism and encoding to replace the one way operation of RFC-5277
- a set of common, transport agnostic message header objects.
- how to bundle multiple event records into a single notification message.
- how to ensure these new capabilities are only used with capable receivers.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on February 23, 2019.

Table of Contents

1. Introduction

Mechanisms to support subscription to event notifications and yang datastore push are being defined in [I-D.draft-ietf-netconf-subscribed-notifications] and [I-D.ietf-netconf-yang-push]. Work on those documents has shown that notifications described in [RFC7950] section 7.16 could benefit from transport independent headers. Communicating the following

information to receiving applications can be done without explicit linkage to an underlying transport protocol:

- the time information was generated
- the time the information was placed in a message and queued for transport
- a signature to verify authenticity
- the process generating the information
- an originating request correlation
- an ability to bundle information records into one a message
- the ability to check for message loss/reordering

The document describes information elements needed for the functions above. It also provides YANG structures for sending messages containing one or more events and/or update records to a receiver.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The definition of notification is in RFC 7950 [RFC7950]. Publisher, receiver, subscription, and event occurence time are defined in [I-D.draft-ietf-netconf-subscribed-notifications].

3. Header Objects

There are a number of transport independent headers which should have common definition. These include:

- subscription-id: provides a reference into the reason the publisher believed the receiver wishes to be notified of this specific information.
- notification-time: the time an event, datastore update, or other item is recognized and recorded within the publisher.
- notification-id: Identifies the name of the notification, per the YANG notification statement. May also provide the name of a yang-

data statement (whether transporting other types of messages is in scope is tbd).

- observation-domain-id: identifies the publisher process which discovered and recorded the event notification. (note: look to reuse the domains set up with IPFIX.)

- message-time: the time the message was packaged sent to the transport layer for delivery to the receiver.

- signature: allows an application to sign a message so that a receiver can verify the authenticity of the message.

- message-id: for a specific message generator, this identifies a message which includes one or more event records. The message-id increments by one with sequential messages.

- message-generator-id: identifier for the process which created the message. This allows disambiguation of an information source, such as the identification of different line cards sending the messages. Used in conjunction with previous-message-id, this can help find drops and duplications when messages are coming from multiple sources on a device. If there is a message-generator-id in the header, then the previous-message-id MUST be the message-id from the last time that message-generator-id was sent.

4. Encapsulation of Header Objects in Messages

A specific set of well-known objects are of potential use to networking layers prior being interpreted by some receiving application layer process. By exposing this object information as part of a header, and by using standardized object names, it becomes possible for this object information to be leveraged in transit.

The objects defined in the previous section are these well-known header objects. These objects are identified within a dedicated header subtree which leads off a particular transportable message. This allows header objects to be easily be decoupled, stripped, and processed separately.

There are two types of transportable messages: one format is used when there is one notification being encapsulated, and another format used when there are many notifications being bundled into one message.

A receiver which supporting this document MUST be able to handle receipt of either type of message from an publisher. It is possible

that changes between message types can occur without any prior indication.

4.1. One Notification per Message

This section will be re-instated if NETCONF WG members are not comfortable with the efficiency of the solution which can encode many notifications per message described below.

4.2. Many Notifications per Message

While possible in some scenarios, it often inefficient to marshal and transport every notification independently. Instead, scale and processing speed can be improved by placing multiple notifications into one transportable bundle.

The format of this bundle appears in the yata-data tree below, and is more completely defined in the yang module. There are three parts of this bundle:

- a message header describing the marshaling, including information such as when the marshaling occurred
- a list of encapsulated information
- an optional message footer for whole-message signing and message-generator integrity verification.

Within the list of encapsulated notifications, there are also three parts:

- a notification header defining what is in an encapsulated notification
- the actual notification itself
- an optional notification footer for individual notification signing and observation-domain integrity verification.

```
       yang-data message
          +--ro message!
             +--ro message-header
             |  +--ro message-time            yang:date-and-time
             |  +--ro message-id?             uint32
             |  +--ro message-generator-id?   string
             |  +--ro notification-count?     uint16
             +--ro notifications*
             |  +--ro notification-header
             |  |  +--ro notification-time         yang:date-and-time
             |  |  +--ro yang-module?              yang:yang-identifier
             |  |  +--ro yang-notification-name?   notification-type
             |  |  +--ro subscription-id*          uint32
             |  |  +--ro notification-id?          uint32
             |  |  +--ro observation-domain-id?    string
             |  +--ro notification-contents?
             |  +--ro notification-footer!
             |     +--ro signature-algorithm    string
             |     +--ro signature-value        string
             |     +--ro integrity-evidence?    string
             +--ro message-footer!
                +--ro signature-algorithm    string
                +--ro signature-value        string
                +--ro integrity-evidence?    string
```

An XML instance of a message might look like:

```
   <yang-data bundled-message
     xmlns="urn:ietf:params:xml:ns:yang:ietf-notification-messages:1.0">
     <message-header>
       <message-time>
             2017-02-14T00:00:05Z
       </message-time>
       <message-id>
             456
       </message-id>
       <notification-count>
             2
       </notification-count>
     </message-header>
     <notifications>
       <notification>
         <notification-header>
           <notification-time>
               2017-02-14T00:00:02Z
           </notification-time>
           <subscription-id>
               823472
           </subscription-id>
           <yang-module>
               ietf-yang-push
           </yang-module>
           <yang-notification-name>
               push-change-update
           </yang-notification-name>
         </notification-header>
         <notification-contents>
           <push-change-update xmlns=
             "urn:ietf:params:xml:ns:yang:ietf-yang-push:1.0">
             <datastore-changes-xml>
               <alpha xmlns="http://example.com/sample-data/1.0">
                  <beta urn:ietf:params:xml:ns:netconf:base:1.0:
                      operation="delete"/>
               </alpha>
             </datastore-changes-xml>
           </push-change-update>
         </notification-contents>
       </notification>
       <notification>
             ...(record #2)...
       </notification>
     </notifications>
   </yang-data>
```

5. Configuration of Headers

A publisher MUST select the set of headers to use within any particular message. The two mandatory headers which MUST always be applied are 'message-time' and 'subscription-id'

Beyond these two mandatory headers, additional headers MAY be included. Configuration of what these optional headers should be can come from the following sources:

1. Publisher wide default headers for all notifications. These are included if an optional header is inserted into 'additional-headers' leaf-list shown in the yang tree below.

2. More notification specific headers may also be desired. If new headers are needed for a specific type of YANG notification, these can be populated through 'additional-notification-headers' leaf-list.

3. An application process may also identify common headers to use when transporting notifications for a specific subscription. How these are identified to a publisher is out-of-scope.

The set of headers used for any particular message is the superset of headers for the items listed above.

The YANG tree showing elements of configuration is depicted in the following figure.

```
module: ietf-notification-messages
    +--rw additional-default-headers {publisher}?
       +--rw additional-headers*                          optional-header
       +--rw yang-notification-specific-default*
          |                          [yang-module yang-notification-name]
          +--rw yang-module                            yang:yang-identifier
          +--rw yang-notification-name                    notification-type
          +--rw additional-notification-headers*
                                              optional-notification-header
```

Configuration Model structure

Of note in this tree is the optional feature of 'publisher'. This feature indicates an ability to send notifications. A publisher supporting this specification MUST also be able to parse any messages received as defined in this document.

6. Discovering Receiver Support

We need capability exchange from the receiver to the publisher at transport session initiation to indicate support for this specification.

For all types of transport connections, if the receiver indicates support for this specification, then it MAY be used. In addition, [RFC5277] one-way notifications MUST NOT be used if the receiver indicates support for this specification to a publisher which also supports it.

Where NETCONF transport is used, advertising this specification's namespace during an earlier client capabilities discovery phase MAY be used to indicate support for this specification:

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

NOTE: It is understood that even though it is allowed in [RFC6241] section 8.1, robust NETCONF client driven capabilities exchange is not something which is common in implementation. Therefore reviewers are asked to submit alternative proposals to the mailing list.

For RESTCONF, a mechanism for capability discovery is TBD. Proposals are also welcome here.

The mechanism described above assumes that a capability discovery phase happens before a subscription is started. This is not always the case. As an example, consider HTTP2 configured subscriptions from section 3.1.3 of [I-D.draft-ietf-netconf-restconf-notif], there is no guarantee that a capability exchange has taken place before the updates are pushed. A solution for this could be that a receiver would reply "ok" and reply with the client capabilities as part of the POST. (Or just use a different HTTP status code like 202 instead of 200 'ok'). As such a requirement creates a new dependency for [I-D.draft-ietf-netconf-restconf-notif] upon this specification, more discussion is required to decide if this is a viable solution.

7. YANG Module

```
<CODE BEGINS> file "ietf-notification-messages@2018-01-31.yang"

module ietf-notification-messages {
  yang-version 1.1;
  namespace
     "urn:ietf:params:xml:ns:yang:ietf-notification-messages";
  prefix nm;

  import ietf-yang-types { prefix yang; }
  import ietf-restconf   { prefix rc;   }


  organization "IETF";
  contact
    "WG Web:   <http://tools.ietf.org/wg/netconf/>
     WG List:  <mailto:netconf@ietf.org>

     Editor:   Eric Voit
               <mailto:evoit@cisco.com>

     Editor:   Henk Birkholz
               <mailto:henk.birkholz@sit.fraunhofer.de>

     Editor:   Alexander Clemm
               <mailto:ludwig@clemm.org>

     Editor:   Andy Bierman
               <mailto:andy@yumaworks.com>

     Editor:   Tim Jenkins
               <mailto:timjenki@cisco.com>";

  description
    "This module contains conceptual YANG specifications for yang-data
    messages carrying notifications with well known header objects.";

  revision 2018-01-31 {
    description
      "Initial version.";

    reference
      "draft-ietf-netconf-notification-messages-03";
  }

 /*
  * FEATURES
```

```
  */

  feature publisher {
    description
      "This feature indicates that support for both publisher and
      receiver of messages complying to the specification.";
  }

  /*
   * IDENTITIES
   */

   /* Identities for common headers */

  identity common-header {
    description
      "A well known header which can be included somewhere within a
      message.";
  }

  identity message-time {
    base common-header;
    description
      "Header information consisting of time the message headers were
      placed generated prior to being sent to transport";
  }

  identity subscription-id {
    base common-header;
    description
      "Header information consisting of the identifier of the
      subscription associated with the notification being
      encapsulated.";
  }

  identity notification-count {
    base common-header;
    description
      "Header information consisting of the quantity of notifications in
      a bundled-message for a specific receiver.";
  }

  identity optional-header {
    base common-header;
    description
      "A well known header which an application may choose to include
      within a message.";
  }
```

```
  identity message-id {
    base optional-header;
    description
      "Header information that identifies a message to a specific
      receiver";
  }

  identity message-generator-id {
    base optional-header;
    description
      "Header information consisting of an identifier for a software
      entity which created the message (e.g., linecard 1).";
  }

  identity message-signature {
    base optional-header;
    description
      "Identifies two elements of header information consisting of a
      signature and the signtature type for the contents of a message.
      Signatures can be useful for originating applications to
      verify record contents even when shipping over unsecure
      transport.";
  }

  identity message-integrity-evidence {
    base optional-header;
    description
      "Header information consisting of the information which backs up
      the assertions made as to the validity of the information
      provided within the message.";
  }


  identity optional-notification-header {
    base optional-header;
    description
      "A well known header which an application may choose to include
      within a message.";
  }

  identity notification-time  {
    base optional-notification-header;
    description
      "Header information consisting of the time an originating process
      created the notification.";
  }

  identity notification-id {
```

```
    base optional-notification-header;
    description
      "Header information consisting of an identifier for an instance
      of a notification egressing a publisher. ";
  }

  identity observation-domain-id {
    base optional-notification-header;
    description
      "Header information identifying the software entity which created
      the notification (e.g., process id).";
  }

  identity notification-signature {
    base optional-notification-header;
    description
      "Header information consisting of the information which backs up
      the assertions made as to the validity of the information
      provided within the notification.";
  }

  identity notification-integrity-evidence {
    base optional-notification-header;
    description
      "Header information consisting of the information which backs up
      the assertions made as to the validity of the information
      provided within the notification.";
  }



  /*
   * TYPEDEFs
   */

  typedef optional-header {
    type identityref {
      base optional-header;
    }
    description
      "Type of header object which may be included somewhere within a
      message.";
  }

  typedef optional-notification-header {
    type identityref {
      base optional-notification-header;
    }
```

```
    description
      "Type of header object which may be included somewhere within a
      message.";
  }

  typedef notification-type {
    type string {
      pattern '[a-zA-Z_][a-zA-Z0-9\-_.]*';
    }
    description
      "The name of a notification within a YANG module.";
    reference
      "RFC-7950 Section 7.16";
  }

  /*
   * GROUPINGS
   */

  grouping message-header {
    description
      "Header information included with a message.";
    leaf message-time {
      type yang:date-and-time;
      mandatory true;
      description
        "time the message was generated prior to being sent to
        transport.";
    }
    leaf message-id {
      type uint32;
      description
        "Id for a message going to a receiver from a message
        generator.  The id will increment by one with each message sent
        from a particular message generator, allowing the message-id
        to be used as a sequence number.";
    }
    leaf message-generator-id {
      type string;
      description
        "Software entity which created the message (e.g., linecard 1).
         The combination of message-id and message-generator-id must be
         unique until reset or a roll-over occurs.";
    }
    leaf notification-count {
      type uint16;
      description
        "Quantity of notification records in a bundled-message
```

```
        specific receiver.";
    }
  }

  grouping notification-within-a-module {
    description
      "A location of a notification within a yang model.";
    leaf yang-module {
      type yang:yang-identifier;
      description
        "Name of the YANG module supported by the publisher.";
    }
    leaf yang-notification-name {
      type notification-type;
       description
         "The name of a notification likely from a YANG module.  Note
         that this object should be in the notification contents, so a
         debate is needed whether this is redundant.";
    }
  }

  grouping notification-header {
    description
      "Common informational objects which might help a receiver
      interpret the meaning, details, or importance of a notification.";
    leaf notification-time {
      type yang:date-and-time;
      mandatory true;
      description
        "Time the system recognized the occurrence of an event.";
    }
    uses notification-within-a-module;
    leaf-list subscription-id {
      type uint32;
      description
        "Id of the subscription which led to the notification being
        generated.";
    }
    leaf notification-id {
      type uint32;
      description
        "Identifier for the notification record.";
    }
    leaf observation-domain-id {
      type string;
      description
        "Software entity which created the notification record (e.g.,
        process id).";
```

```
    }
  }

  grouping security-footer {
    description
      "Reusable grouping for common objects which apply to the the
      signing of notifications or messages.";
    leaf signature-algorithm {
      type string;
      mandatory true;
      description
        "The technology with which an originator signed of some
        delineated contents.";
    }
    leaf signature-value {
      type string;
      mandatory true;
      description
        "Any originator signing of the contents of a header and
        content.  This is useful for verifying contents even when
        shipping over unsecure transport.";
    }
    leaf integrity-evidence {
      type string;
      description
        "This mechanism allows a verifier to ensure that the use of the
        private key, represented by the corresponding public key
        certificate, was performed with a TCG compliant TPM
        environment.  This evidence is never included in within any
        signature.";
      reference
        "TCG Infrastructure Workgroup, Subject Key Attestation Evidence
        Extension, Specification Version 1.0, Revision 7.";
    }
  }


  /*
   * YANG-DATA messages for receivers
   */


   rc:yang-data message {
    container message {
      presence
        "Indicates attempt to communicate notifications to a receiver.";
      description
        "Message to a receiver containing one or more notifications";
```

```
      container message-header {
        description
          "Header info for messages.";
        uses message-header;
      }
      list notifications {
        description
          "Set of notifications to a receiver.";
        container notification-header {
          description
            "Header info for a notification.";
          uses notification-header;
        }
        anydata notification-contents {
          description
            "Encapsulates objects following YANG's notification-stmt
            grammar of RFC-7950 section 14.  Within are the notified
            objects the publisher actually generated in order to be
            passed to a receiver after all filtering has completed.";
        }
        container notification-footer {
          presence
            "Indicates attempt to secure a notification.";
          description
            "Signature and evidence for messages.";
          uses security-footer;
        }
      }
      container message-footer {
        presence
          "Indicates attempt to secure the entire message.";
        description
          "Signature and evidence for messages.";
        uses security-footer;
      }
    }
  }

  /*
   * DATA-NODES
   */

  container additional-default-headers {
    if-feature "publisher";
    description
      "This container maintains a list of which additional notifications
      should use which optional headers if the receiver supports this
      specification.";
```

```
    leaf-list additional-headers {
      type optional-header;
      description
        "This list contains the identities of the optional header types
        which are to be included within each message from this
        publisher.";
    }
    list yang-notification-specific-default {
      key "yang-module yang-notification-name";
      description
        "For any included YANG notifications, this list provides
        additional optional headers which should be placed within the
        container notification-header if the receiver supports this
        specification.  This list incrementally adds to any headers
        indicated within the leaf-list 'additional-headers'.";
      uses notification-within-a-module;
      leaf-list additional-notification-headers {
         type optional-notification-header;
         description
           "The set of additional default headers which will be sent
           for a specific YANG notification.";
      }
    }
  }
}

<CODE ENDS>
```

8. Backwards Compatibility

With this specification, there is no change to YANG's 'notification' statement

Legacy clients are unaffected, and existing users of [RFC5277], [RFC7950], and [RFC8040] are free to use current behaviors until all involved device support this specification.

9. Security Considerations

Certain headers might be computationally complex for a publisher to deliver. Signatures or encryption are two examples of this. It MUST be possible to suspend or terminate a subscription due to lack of resources based on this reason.

Decisions on whether to bundle or not to a receiver are fully under the purview of the Publisher. A receiver could slow delivery to existing subscriptions by creating new ones. (Which would result in the publisher going into a bundling mode.)

10. Acknowledgments

For their valuable comments, discussions, and feedback, we wish to acknowledge Martin Bjorklund, Einar Nilsen-Nygaard, and Kent Watsen.

11. References

11.1. Normative References

[I-D.draft-ietf-netconf-subscribed-notifications]
Voit, E., Clemm, A., Gonzalez Prieto, A., Tripathy, A., and E. Nilsen-Nygaard, "Custom Subscription to Event Streams", draft-ietf-netconf-subscribed-notifications-16 (work in progress), August 2018.

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

[RFC5277] Chisholm, S. and H. Trevino, "NETCONF Event Notifications", RFC 5277, DOI 10.17487/RFC5277, July 2008, <https://www.rfc-editor.org/info/rfc5277>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

11.2. Informative References

[I-D.draft-ietf-netconf-restconf-notif]
Voit, Eric., Clemm, Alexander., Tripathy, A., Nilsen-Nygaard, E., and Alberto. Gonzalez Prieto, "Restconf and HTTP transport for event notifications", June 2018, <https://datatracker.ietf.org/doc/draft-ietf-netconf-restconf-notif/>.

[I-D.ietf-netconf-yang-push]
Clemm, Alexander., Voit, Eric., Gonzalez Prieto, Alberto., Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "YANG Datastore Subscription", August 2018, <https://datatracker.ietf.org/doc/draft-ietf-netconf-yang-push/>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

Appendix A. Changes between revisions

(To be removed by RFC editor prior to publication)

v03 - v04

- Terminology tweaks.
- Revision before expiration. Awaiting closure of SN prior to update.

v02 - v03

- Removed the option for an unbundled message. This might be re-added later for transport efficiency if desired by the WG
- New message structure driven by the desire to put the signature information at the end.

v01 - v02

- Fixed the yang-data encapsulation container issue
- Updated object definitions to point to RFC-7950 definitions
- Added headers for module and notification-type.

v00 - v01

- Alternative to 5277 one-way notification added
- Storage of default headers by notification type
- Backwards compatibility
- Capability discovery
- Move to yang-data

- Removed dscp and record-type as common headers. (Record type can be determined by the namespace of the record contents. Dscp is useful where applications need internal communications within a Publisher, but it is unclear as to whether this use case need be exposed to a receiver.

Appendix B. Issues being worked

(To be removed by RFC editor prior to publication)

Is this capability just for notifications, or is it for any yang-data element too?

A complete JSON document is supposed to be sent as part of Media Type "application/yang-data+json". As we are sending separate notifications after each other, we need to choose whether we start with some extra encapsulation for the very first message pushed, or if we want a new Media Type for streaming updates.

Improved discovery mechanisms for NETCONF

Should we defer support for HTTP2 configured subscriptions until this draft is available? Without capabilities exchange, it might just be easier to wait. In addition, JSON encoding still needs a notification type which is not exising or represented in referenceable in existing yang-models.

Need to ensure the proper references exist to a notification definition driven by RFC-7950 which is acceptable to other eventual users of this specification.

We need to link to Andy Bierman's anydata extensibility draft for informational purposes. This is under a WG adoption call.

Appendix C. Subscription Specific Headers

(To be removed by RFC editor prior to publication)

This section discusses a future functional addition which could leverage this draft. It is included for informational purposes only.

A dynamic subscriber might want to mandate that certain headers be used for push updates from a publisher. Some examples of this include a subscriber requesting to:

- establish this subscription, but just if transport messages containing the pushed data will be encrypted,

- establish this subscription, but only if you can attest to the information being delivered in requested notification records, or

- provide a sequence-id for all messages to this receiver (in order to check for loss).

Providing this type of functionality would necessitate a new revision of the [I-D.draft-ietf-netconf-subscribed-notifications]'s RPCs and state change notifications. Subscription specific header information would overwrite the default headers identified in this document.

Appendix D. Implications to Existing RFCs

(To be removed by RFC editor prior to publication)

YANG one-way exchanges currently use a non-extensible header and encoding defined in section 4 of RFC-5277. These RFCs MUST be updated to enable this draft. These RFCs SHOULD be updated to provide examples

D.1. Implications to RFC-7950

Sections which expose netconf:capability:notification:1.0 are 4.2.10

Sections which provide examples using netconf:notification:1.0 are 7.10.4, 7.16.3, and 9.9.6

D.2. Implications to RFC-8040

Section 6.4 demands use of RFC-5277's netconf:notification:1.0, and later in the section provides an example.

Authors' Addresses

Eric Voit
Cisco Systems

Email: evoit@cisco.com

Henk Birkholz
Fraunhofer SIT

Email: henk.birkholz@sit.fraunhofer.de

Andy Bierman
YumaWorks

Email: andy@yumaworks.com

Alexander Clemm
Huawei

Email: ludwig@clemm.org

Tim Jenkins
Cisco Systems

Email: timjenki@cisco.com