NETCONF
Internet-Draft
Intended status: Standards Track
Expires: March 16, 2019

E. Voit
R. Rahman
E. Nilsen-Nygaard
Cisco Systems
A. Clemm
Huawei
A. Bierman
YumaWorks

September 12, 2018

RESTCONF Transport for Event Notifications
draft-ietf-netconf-restconf-notif-07

Abstract

This document defines a RESTCONF binding to the dynamic subscription capability of both subscribed notifications and YANG-Push. Subscriptions to publisher defined event streams or nodes/subtrees of YANG Datastores is supported.

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 March 16, 2019.

Table of Contents

1. Introduction

Mechanisms to support event subscription and push are defined in [I-D.draft-ietf-netconf-subscribed-notifications]. Enhancements to [I-D.draft-ietf-netconf-subscribed-notifications] which enable YANG datastore subscription and push are defined in [I-D.ietf-netconf-yang-push]. This document provides a transport specification for dynamic subscriptions over RESTCONF [RFC8040]. Driving these requirements is [RFC7923].

The streaming of notifications encapsulating the resulting information push can be done with either HTTP1.1 [RFC7231] or HTTP2 [RFC7540].

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

The following terms use the definitions from [I-D.draft-ietf-netconf-subscribed-notifications]: dynamic subscription, event stream, notification message, publisher, receiver, subscriber, and subscription.

Other terms reused include datastore, which is defined in [RFC8342], and HTTP2 stream which maps to the definition of "stream" within [RFC7540], Section 2.

[note to the RFC Editor - please replace XXXX within this document with the number of this document]

3. Dynamic Subscriptions

This section provides specifics on how to establish and maintain dynamic subscriptions over HTTP 1.1 and HTTP2 via signaling messages transported over RESTCONF [RFC8040]. Subscribing to event streams is accomplished in this way via a RESTCONF POST into RPCs defined within [I-D.draft-ietf-netconf-subscribed-notifications] Section 2.4. YANG datastore subscription is accomplished via augmentations to [I-D.draft-ietf-netconf-subscribed-notifications] as described within [I-D.ietf-netconf-yang-push] Section 4.4.

Common across all HTTP based dynamic subscriptions is that a POST needs to be made against a specific URI on the Publisher. Subscribers cannot pre-determine the URI against which a subscription might exist on a publisher, as the URI will only exist after the "establish-subscription" has been accepted. The subscription URI will be determined and sent as part of the response to the "establish-subscription", and a subsequent POST to this URI will be done in order to start the flow of notification messages back to the subscriber. A subscription does not move to the active state as per Section 2.4.1. of [I-D.draft-ietf-netconf-subscribed-notifications] until the POST is received.

3.1. Transport Connectivity

For a dynamic subscription, where an HTTP client session doesn't already exist, a new client session is initiated from the subscriber. If the subscriber is unsure if HTTP2 is supported by the publisher, HTTP1.1 will be used for initial messages, and these messages will include an HTTP version upgrade request as per [RFC7230], Section 6.7. If a publisher response indicates that HTTP2 is supported, HTTP2 will be used between subscriber and publisher for future HTTP interactions as per [RFC7540].

A subscriber SHOULD establish the HTTP session over TLS [RFC5246] in order to secure the content in transit.

Without the involvement of additional protocols, neither HTTP1.1 nor HTTP2 sessions by themselves allow for a quick recognition of when the communication path has been lost with the publisher. Where quick recognition of the loss of a publisher is required, a subscriber SHOULD connect over TLS [RFC5246], and use a TLS heartbeat [RFC6520] to track HTTP session continuity. In the case where a TLS heartbeat is included, it should be sent just from receiver to publisher. Loss of the heartbeat MUST result in any subscription related TCP sessions between those endpoints being torn down. A subscriber can then attempt to re-establish.

3.2. Discovery

Subscribers can learn what event streams a RESTCONF server supports by querying the "streams" container of ietf-subscribed-notification.yang. Subscribers can learn what datastores a RESTCONF server supports by following [I-D.draft-ietf-netconf-nmda-restconf].

3.3. RESTCONF RPCs and HTTP Status Codes

Specific HTTP responses codes as defined in [RFC7231] section 6 will indicate the result of RESTCONF RPC requests with publisher. An HTTP status code of 200 is the proper response to any successful RPC defined within [I-D.draft-ietf-netconf-subscribed-notifications] or [I-D.ietf-netconf-yang-push].

If a publisher fails to serve the RPC request for one of the reasons indicated in [I-D.draft-ietf-netconf-subscribed-notifications] Section 2.4.6 or [I-D.ietf-netconf-yang-push] Appendix A, this will be indicated by "406" status code transported in the HTTP response.

When a "406" status code is returned, the RPC reply MUST include an "rpc-error" element per [RFC8040] Section 7.1 with the following parameter values:

- an "error-type" node of "application".

- an "error-tag" node of "operation-failed".

- an "error-app-tag" node with the value being a string that corresponds to an identity associated with the error, as defined in [I-D.draft-ietf-netconf-subscribed-notifications] section 2.4.6 for general subscriptions, and [I-D.ietf-netconf-yang-push] Appendix A.1, for datastore subscriptions. The tag to use depends on the RPC for which the error occurred. Viable errors for different RPCs are as follows:

RPC	select an identity with a base
establish-subscription	establish-subscription-error
modify-subscription	modify-subscription-error
delete-subscription	delete-subscription-error
kill-subscription	kill-subscription-error
resynch-subscription	resynch-subscription-error

Each error identity will be inserted as the "error-app-tag" using JSON encoding following the form <modulename>:<identityname>. An example of such as valid encoding would be "ietf-subscribed-notifications:no-such-subscription".

- In case of error responses to an "establish-subscription" or "modify-subscription" request there is the option of including an "error-info" node. This node may contain hints for parameter settings that might lead to successful RPC requests in the future. Following are the yang-data structures which may be returned:

establish-subscription	returns hints in yang-data structure
target: event stream	establish-subscription-stream-error-info
target: datastore	establish-subscription-datastore-error-info

modify-subscription	returns hints in yang-data structure
target: event stream	modify-subscription-stream-error-info
target: datastore	modify-subscription-datastore-error-info

The yang-data included within "error-info" SHOULD NOT include the optional leaf "error-reason", as such a leaf would be redundant with information that is already placed within the "error-app-tag".

In case of an rpc error as a result of a "delete-subscription", a "kill-subscription", or a "resynch-subscription" request, no "error-info" needs to be included, as the "subscription-id" is the only RPC input parameter and no hints regarding this RPC input parameters need to be provided.

Note that "error-path" does not need to be included with the "rpc-error" element, as subscription errors are generally not associated with nodes in the datastore but with the choice of RPC input parameters.

3.4. Call Flow for HTTP2

Requests to [I-D.draft-ietf-netconf-subscribed-notifications] or [I-D.ietf-netconf-yang-push] augmented RPCs are sent on one or more HTTP2 streams indicated by (a) in Figure 1. A successful "establish-subscription" will result in an RPC response returned with both a subscription identifier which uniquely identifies a subscription, as well as a URI which uniquely identifies the location of subscription on the publisher. This URI is defined via the "uri" leaf the Data Model in Section 8.

An HTTP POST is then sent on a logically separate HTTP2 stream (b) to the URI on the publisher. This initiates to initiate the flow of notification messages which are sent in HTTP Data frames as a response to the POST. In the case below, a newly established subscription has its associated notification messages pushed over HTTP2 stream (7). These notification messages are placed into a HTTP2 Data frame (see [RFC7540] Section 6.1).

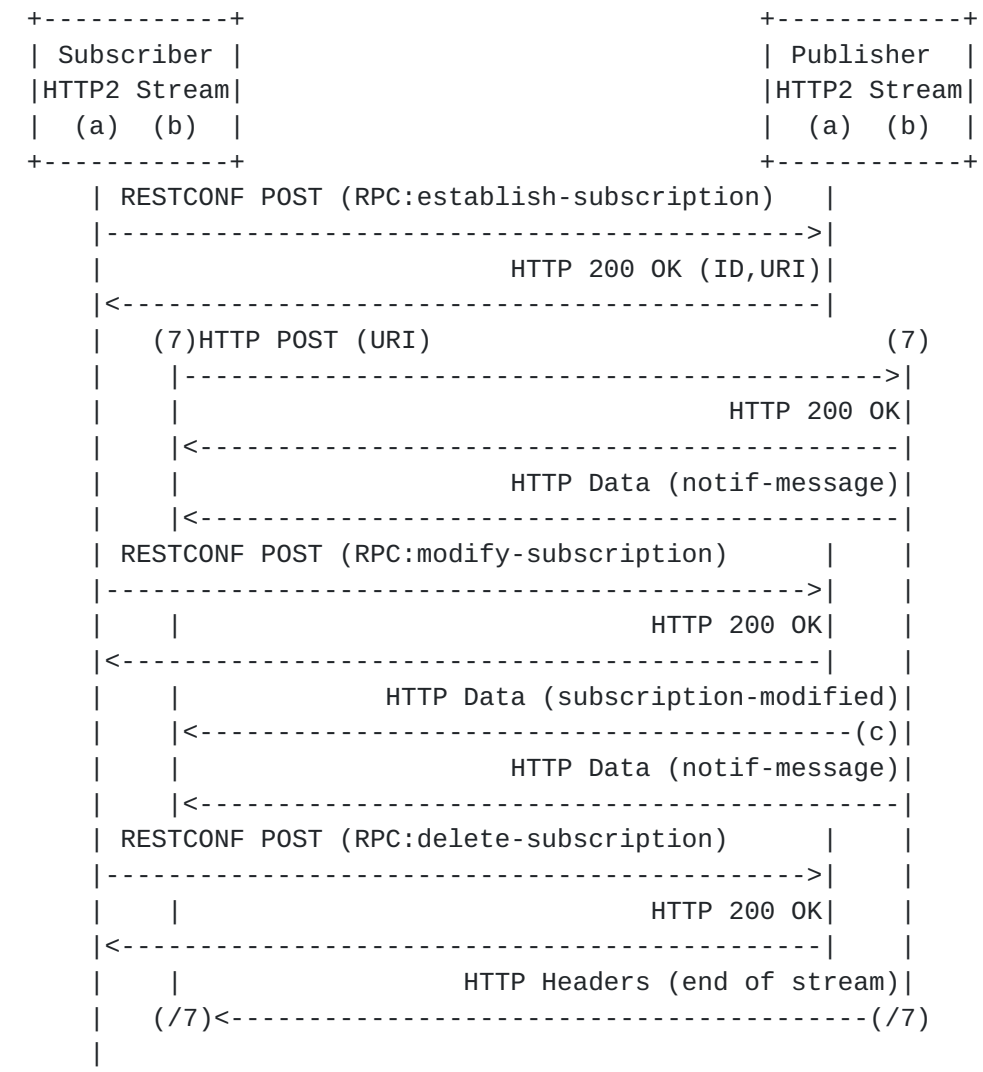

Figure 1: Dynamic with HTTP2

Additional requirements for dynamic subscriptions over HTTP2 include:

- A unique HTTP2 stream MAY be used for each subscription.

- A single HTTP2 stream MUST NOT be used for subscriptions with different DSCP values.

- All subscription state notifications from a publisher MUST be returned in a separate HTTP Data frame within the HTTP2 stream used by the subscription to which the state change refers.

- In addition to an RPC response for a "modify-subscription" RPC traveling over (a), a "subscription-modified" state change notification must be sent within HTTP2 stream (b). This allows the receiver to know exactly when the new terms of the

subscription have been applied to the notification messages. See arrow (c).

- Additional RPCs for a particular subscription MUST NOT use the HTTP2 stream currently providing notification messages subscriptions.

- An HTTP end of stream message MUST not be sent until all subscriptions using that HTTP2 stream have completed.

3.5. Call flow for HTTP1.1

The call flow is defined in Figure 2. Requests to [I-D.draft-ietf-netconf-subscribed-notifications] or [I-D.ietf-netconf-yang-push] augmented RPCs are sent on a TCP connection indicated by (a). A successful "establish-subscription" will result in an RPC response returned with both a subscription identifier which uniquely identifies a subscription, as well as a URI which uniquely identifies the location of subscription on the publisher (b). This URI is defined via the "uri" leaf the Data Model in Section 8.

An HTTP POST is then sent on a logically separate TCP connection (b) to the URI on the publisher. This initiates to initiate the flow of notification messages which are sent in SSE [W3C-20150203] as a response to the POST.

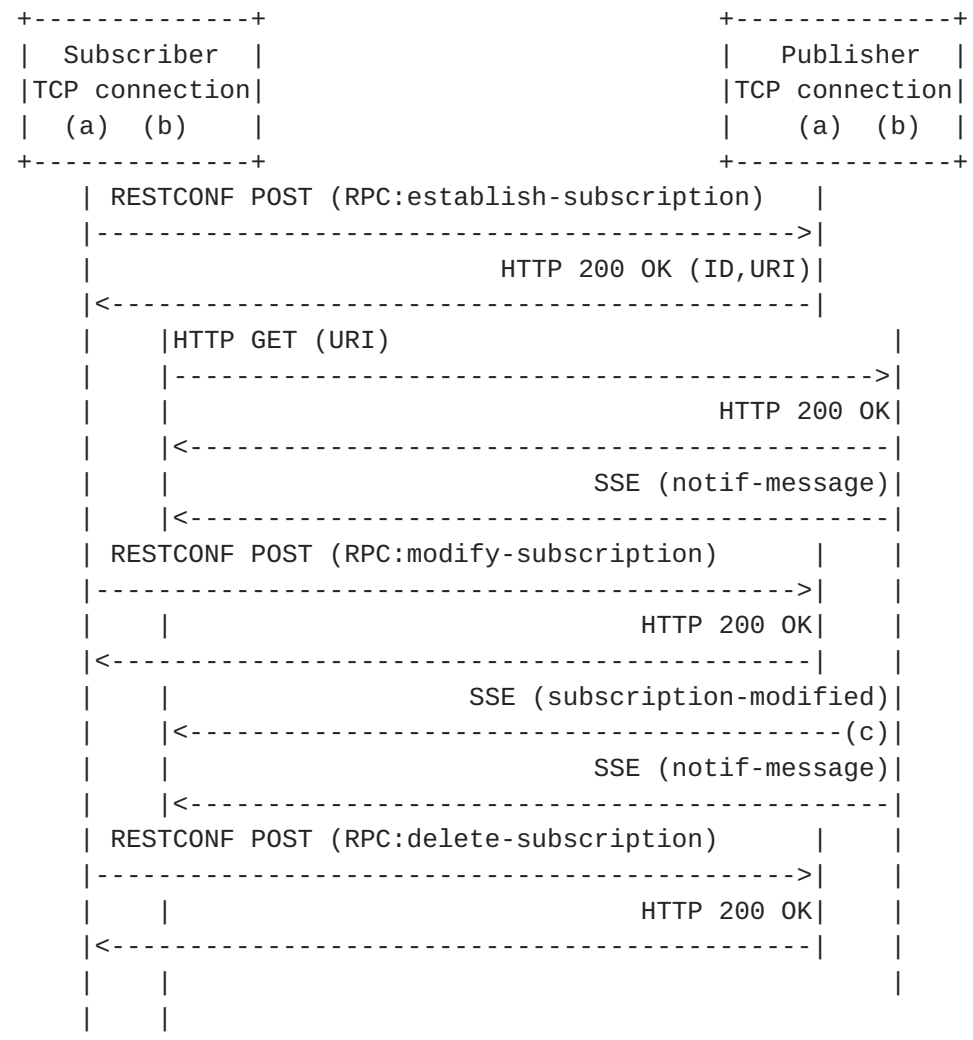

Figure 2: Dynamic with HTTP1.1

Additional requirements for dynamic subscriptions over HTTP1.1 include:

- All subscription state notifications from a publisher MUST be returned in a separate SSE message used by the subscription to which the state change refers.

- Subscription RPCs MUST NOT use the TCP connection currently providing notification messages for that subscription.

- In addition to an RPC response for a "modify-subscription" RPC traveling over (a), a "subscription-modified" state change notification must be sent within stream (b). This allows the receiver to know exactly when the new terms of the subscription have been applied to the notification messages. See arrow (c).

Open question, should we just eliminate this possibility of HTTP1.1 for subscriptions? It would make the design simpler.

4. QoS Treatment

To meet subscription quality of service promises, the publisher MUST take any existing subscription "dscp" and apply it to the DSCP marking in the IP header.

In addition, where HTTP2 transport is available to a notification message queued for transport to a receiver, the publisher MUST:

- take any existing subscription "priority" and copy it into the HTTP2 stream priority, and

- take any existing subscription "dependency" and map the HTTP2 stream for the parent subscription into the HTTP2 stream dependency.

5. Mandatory JSON and datastore support

A publisher supporting [I-D.ietf-netconf-yang-push] MUST support the "operational" datastore as defined by [RFC8342].

The "encode-json" feature of [I-D.draft-ietf-netconf-subscribed-notifications] is mandatory to support. This indicates that JSON is a valid encoding for RPCs, state change notifications, and subscribed content.

6. Notification Messages

Notification messages transported over HTTP will be encoded using one-way operation schema defined within [RFC5277], section 4.

7. YANG Tree

The YANG model defined in Section 8 has one leaf augmented into four places of [I-D.draft-ietf-netconf-subscribed-notifications], plus two identities. As the resulting full tree is large, it will only be inserted at later stages of this document.

8. YANG module

This module references [I-D.draft-ietf-netconf-subscribed-notifications].

```
<CODE BEGINS> file "ietf-restconf-subscribed-notifications@2018-09-12.yang"
module ietf-restconf-subscribed-notifications {
```

```
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-restconf-subscribed-notifications";

  prefix rsn;

  import ietf-subscribed-notifications {
    prefix sn;
  }
  import ietf-inet-types {
    prefix inet;
  }

  organization "IETF NETCONF (Network Configuration) Working Group";
  contact
    "WG Web:   <http:/tools.ietf.org/wg/netconf/>
     WG List:  <mailto:netconf@ietf.org>

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

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

     Editor:   Reshad Rahman
               <mailto:rrahman@cisco.com>";

  description
    "Defines RESTCONF as a supported transport for subscribed
    event notifications.

    Copyright (c) 2018 IETF Trust and the persons identified as authors
    of the code.  All rights reserved.

    Redistribution and use in source and binary forms, with or without
    modification, is permitted pursuant to, and subject to the license
    terms contained in, the Simplified BSD License set forth in Section
    4.c of the IETF Trust's Legal Provisions Relating to IETF Documents
    (https://trustee.ietf.org/license-info).

    This version of this YANG module is part of RFC XXXX; see the RFC
    itself for full legal notices.";

  revision 2018-09-12 {
    description
      "Initial version";
    reference
      "RFC XXXX: RESTCONF Transport for Event Notifications";
```

```
  }

  grouping uri {
    description
      "Provides a reusable description of a URI.";
    leaf uri {
      type inet:uri;
      config false;
      description
        "Location of a subscription specific URI on the publisher.";
    }
  }

  augment "/sn:establish-subscription/sn:output" {
    description
      "This augmentation allows HTTP specific parameters for a
      response to a publisher's subscription request.";
    uses uri;
  }

  augment "/sn:subscriptions/sn:subscription" {
    description
      "This augmentation allows HTTP specific parameters to be
       exposed for a subscription.";
    uses uri;
  }

  augment "/sn:subscription-started" {
    description
      "This augmentation allows HTTP specific parameters to be included
      part of the notification that a subscription has started.";
    uses uri;
  }

  augment "/sn:subscription-modified" {
    description
      "This augmentation allows HTTP specific parameters to be included
      part of the notification that a subscription has been modified.";
    uses uri;
  }

}
<CODE ENDS>
```

9. IANA Considerations

This document registers the following namespace URI in the "IETF XML Registry" [RFC3688]:

URI: urn:ietf:params:xml:ns:yang:ietf-restconf-subscribed-notifications
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.

This document registers the following YANG module in the "YANG Module Names" registry [RFC6020]:

Name: ietf-restconf-subscribed-notifications
Namespace: urn:ietf:params:xml:ns:yang:ietf-restconf-subscribed-notifications
Prefix: rsn
Reference: RFC XXXX: RESTCONF Transport for Event Notifications

10. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management transports such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC5246].

The one new data node introduced in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to this data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

Container: "/subscriptions"

- "uri": leaf will show where subscribed resources might be located on a publisher. Access control must be set so that only someone with proper access permissions, and perhaps even HTTP session has the ability to access this resource.

11. Acknowledgments

We wish to acknowledge the helpful contributions, comments, and suggestions that were received from: Ambika Prasad Tripathy, Alberto Gonzalez Prieto, Susan Hares, Tim Jenkins, Balazs Lengyel, Kent Watsen, Michael Scharf, and Guangying Zheng.

12. References

12.1. 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-13 (work in progress), April 2018.

[I-D.ietf-netconf-yang-push]
Clemm, A., Voit, E., Gonzalez Prieto, A., Prasad Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "Subscribing to YANG datastore push updates", March 2017, <https://datatracker.ietf.org/doc/draft-ietf-netconf-yang-push/>.

[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>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

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

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

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

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC6520] Seggelmann, R., Tuexen, M., and M. Williams, "Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension", RFC 6520, DOI 10.17487/RFC6520, February 2012, <https://www.rfc-editor.org/info/rfc6520>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/info/rfc7230>.

[RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <https://www.rfc-editor.org/info/rfc7540>.

[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>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[W3C-20150203] "Server-Sent Events, World Wide Web Consortium CR CR-eventsource-20121211", February 2015, <https://www.w3.org/TR/2015/REC-eventsource-20150203/>.

12.2. Informative References

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

[I-D.draft-ietf-netconf-nmda-restconf] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "RESTCONF Extensions to Support the Network Management Datastore Architecture", April 2018, <https://datatracker.ietf.org/doc/draft-ietf-netconf-nmda-restconf/>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <https://www.rfc-editor.org/info/rfc7231>.

[RFC7923] Voit, E., Clemm, A., and A. Gonzalez Prieto, "Requirements for Subscription to YANG Datastores", RFC 7923, DOI 10.17487/RFC7923, June 2016, <https://www.rfc-editor.org/info/rfc7923>.

Appendix A. Examples

This section is non-normative. To allow easy comparison, this section mirrors the functional examples shown with NETCONF over XML within [I-D.draft-ietf-netconf-netconf-event-notifications]. In addition, HTTP2 vs HTTP1.1 headers are not shown as the contents of the JSON encoded objects are identical within.

A.1. Dynamic Subscriptions

A.1.1. Establishing Dynamic Subscriptions

The following figure shows two successful "establish-subscription" RPC requests as per [I-D.draft-ietf-netconf-subscribed-notifications]. The first request is given a subscription identifier of 22, the second, an identifier of 23.

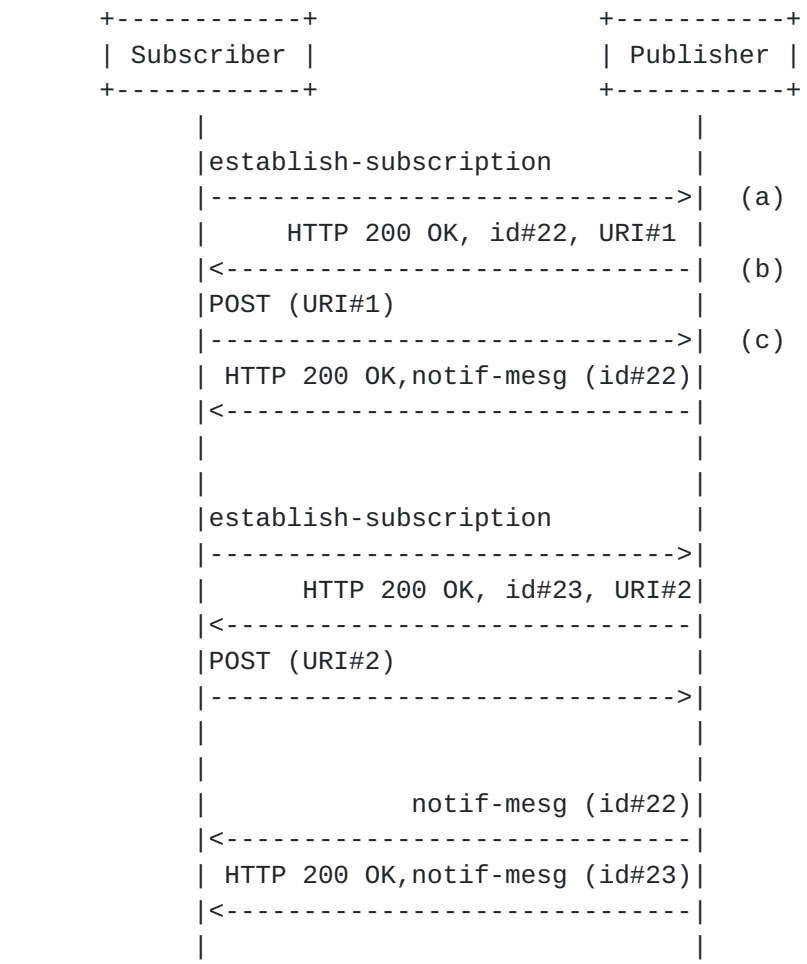

Figure 3: Multiple subscriptions over RESTCONF/HTTP

To provide examples of the information being transported, example messages for interactions in Figure 3 are detailed below:

```
POST /restconf/operations/subscriptions:establish-subscription

{
   "ietf-subscribed-notifications:input": {
      "stream": "NETCONF",
      "stream-xpath-filter": "/ex:foo/",
      "dscp": "10"
   }
}
```

Figure 4: establish-subscription request (a)

As publisher was able to fully satisfy the request, the publisher sends the subscription identifier of the accepted subscription, and the URI:

```
HTTP status code - 200

{
   "id": "22",
   "uri": "/subscriptions/22"
}
```

Figure 5: establish-subscription success (b)

Upon receipt of the successful response, the subscriber POSTs to the provided URI to start the flow of notification messages. When the publisher receives this, the subscription is moved to the active state (c).

```
POST /restconf/operations/subscriptions/22
```

Figure 6: establish-subscription subsequent POST

While not shown in Figure 3, if the publisher had not been able to fully satisfy the request, or subscriber has no authorization to establish the subscription, the publisher would have sent an RPC error response. For instance, if the "dscp" value of 10 asserted by the subscriber in Figure 4 proved unacceptable, the publisher may have returned:

```
      HTTP status code - 406

      { "ietf-restconf:errors" : {
          "error" : [
            {
              "error-type": "application",
              "error-tag": "operation-failed",
              "error-severity": "error",
              "error-app-tag":
                  "ietf-subscribed-notifications:dscp-unavailable"
            }
          ]
        }
      }
```

Figure 7: an unsuccessful establish subscription

The subscriber can use this information in future attempts to establish a subscription.

A.1.2. Modifying Dynamic Subscriptions

An existing subscription may be modified. The following exchange shows a negotiation of such a modification via several exchanges between a subscriber and a publisher. This negotiation consists of a failed RPC modification request/response, followed by a successful one.

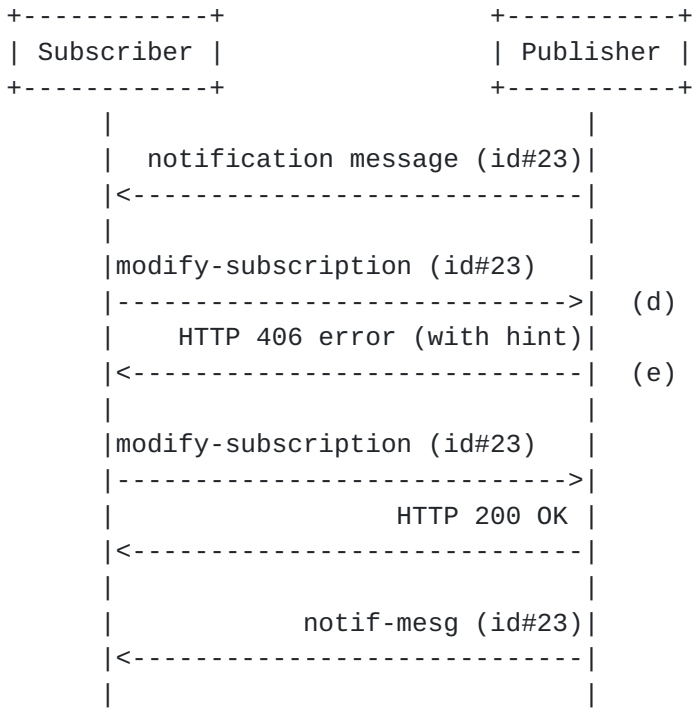

Figure 8: Interaction model for successful subscription modification

If the subscription being modified in Figure 8 is a datastore subscription as per [I-D.ietf-netconf-yang-push], the modification request made in (d) may look like that shown in Figure 9. As can be seen, the modifications being attempted are the application of a new xpath filter as well as the setting of a new periodic time interval.

```
POST /restconf/operations/subscriptions:modify-subscription

{
 "ietf-subscribed-notifications:input": {
    "id": "23",
    "ietf-yang-push:datastore-xpath-filter":
      "/interfaces-state/interface/oper-status"
      "ietf-yang-push:periodic": {
       "ietf-yang-push:period": "500"
      }
 }
}
```

Figure 9: Subscription modification request (c)

If the publisher can satisfy both changes, the publisher sends a positive result for the RPC. If the publisher cannot satisfy either of the proposed changes, the publisher sends an RPC error response (e). The following is an example RPC error response for (e) which includes a hint. This hint is an alternative time period value which might have resulted in a successful modification:

```
      HTTP status code - 406

      { "ietf-restconf:errors" : {
          "error" : [
            "error-type": "application",
            "error-tag": "operation-failed",
            "error-severity": "error",
            "error-app-tag": "ietf-yang-push:period-unsupported",
            "error-info": {
              "ietf-yang-push":
              "modify-subscription-datastore-error-info": {
                 "period-hint": "3000"
              }
            }
          ]
        }
      }
```

Figure 10: Modify subscription failure with Hint (e)

A.1.3. Deleting Dynamic Subscriptions

The following demonstrates deleting a subscription. This subscription may have been to either a stream or a datastore.

```
POST /restconf/operations/subscriptions:delete-subscription

{
 "delete-subscription": {
    "id": "22"
 }
}
```

Figure 11: Delete subscription

If the publisher can satisfy the request, the publisher replies with success to the RPC request.

If the publisher cannot satisfy the request, the publisher sends an error-rpc element indicating the modification didn't work. Figure 12 shows a valid response for existing valid subscription identifier, but that subscription identifier was created on a different transport session:

```
HTTP status code - 406

{
  "ietf-restconf:errors" : {
    "error" : [
      "error-type": "application",
      "error-tag": "operation-failed",
      "error-severity": "error",
      "error-app-tag":
         "ietf-subscribed-notifications:no-such-subscription"
    ]
  }
}
```

Figure 12: Unsuccessful delete subscription

A.2. Subscription State Notifications

A publisher will send subscription state notifications according to the definitions within [I-D.draft-ietf-netconf-subscribed-notifications]).

A.2.1. subscription-started and subscription-modified

A "subscription-started" encoded in JSON would look like:

```
{
  "ietf-restconf:notification" : {
    "eventTime": "2007-09-01T10:00:00Z",
    "ietf-subscribed-notifications:subscription-started": {
      "id": "39",
      "transport": "HTTP2",
      "stream-xpath-filter": "/ex:foo",
      "stream": {
         "ietf-netconf-subscribed-notifications" : "NETCONF"
      }
    }
  }
}
```

Figure 13: subscription-started subscription state notification

The "subscription-modified" is identical to Figure 13, with just the word "started" being replaced by "modified".

A.2.2. subscription-completed, subscription-resumed, and replay-complete

A "subscription-completed" would look like:

```
{
  "ietf-restconf:notification" : {
    "eventTime": "2007-09-01T10:00:00Z",
    "ietf-subscribed-notifications:subscription-completed": {
      "id": "39",
    }
  }
}
```

Figure 14: subscription-completed notification in JSON

The "subscription-resumed" and "replay-complete" are virtually identical, with "subscription-completed" simply being replaced by "subscription-resumed" and "replay-complete".

A.2.3. subscription-terminated and subscription-suspended

A "subscription-terminated" would look like:

```
{
  "ietf-restconf:notification" : {
    "eventTime": "2007-09-01T10:00:00Z",
    "ietf-subscribed-notifications:subscription-terminated": {
      "id": "39",
      "error-id": "suspension-timeout"
    }
  }
}
```

Figure 15: subscription-terminated subscription state notification

The "subscription-suspended" is virtually identical, with "subscription-terminated" simply being replaced by "subscription-suspended".

Appendix B. Changes between revisions

(To be removed by RFC editor prior to publication)

v06 - v07

- Removed configured subscriptions.
- Subscription identifier renamed to id.

v05 - v06

- JSON examples updated by Reshad.

v04 - v05

- Error mechanisms updated to match embedded RESTCONF mechanisms
- Restructured format and sections of document.
- Added a YANG data model for HTTP specific parameters.
- Mirrored the examples from the NETCONF transport draft to allow easy comparison.

v03 - v04

- Draft not fully synched to new version of subscribed-notifications yet.
- References updated

v02 - v03

- Event notification reframed to notification message.
- Tweaks to wording/capitalization/format.

v01 - v02

- Removed sections now redundant with [I-D.draft-ietf-netconf-subscribed-notifications] and [I-D.ietf-netconf-yang-push] such as: mechanisms for subscription maintenance, terminology definitions, stream discovery.
- 3rd party subscriptions are out-of-scope.
- SSE only used with RESTCONF and HTTP1.1 dynamic subscriptions
- Timeframes for event tagging are self-defined.
- Clean-up of wording, references to terminology, section numbers.

v00 - v01

- Removed the ability for more than one subscription to go to a single HTTP2 stream.
- Updated call flows. Extensively.
- SSE only used with RESTCONF and HTTP1.1 dynamic subscriptions
- HTTP is not used to determine that a receiver has gone silent and is not Receiving Event Notifications
- Many clean-ups of wording and terminology

Authors' Addresses

Eric Voit
Cisco Systems

Email: evoit@cisco.com

Reshad Rahman
Cisco Systems

Email: rrahman@cisco.com

Einar Nilsen-Nygaard
Cisco Systems

Email: einarnn@cisco.com

Alexander Clemm
Huawei

Email: ludwig@clemm.org

Andy Bierman
YumaWorks

Email: andy@yumaworks.com