NETCONF
Internet-Draft
Intended status: Standards Track
Expires: April 2, 2017

E. Voit
A. Clemm
A. Gonzalez Prieto
A. Tripathy
E. Nilsen-Nygaard
Cisco Systems
A. Bierman
YumaWorks

September 29, 2016

Restconf and HTTP Transport for Event Notifications
draft-ietf-netconf-restconf-notif-01

Abstract

This document defines Restconf, HTTP2, and HTTP1.1 bindings for the transport Subscription requests and corresponding push updates. Being subscribed may be both Event Notifications and YANG Datastores.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Mechanisms to support Event subscription and push are defined in [rfc5277bis]. Enhancements to [rfc5277bis] which enable YANG Datastore subscription and push are defined in [yang-push]. This document provides a transport specification for these protocols over Restconf and HTTP. Driving these requirements is [RFC7923].

The streaming of Subscription Event Notifications has synergies with HTTP2 streams. Benefits which can be realized when transporting events directly HTTP2 [RFC7540] include:

o Elimination of head-of-line blocking

o Weighting and proportional dequeuing of Events from different subscriptions

- o Explicit precedence in subscriptions so that events from one subscription must be sent before another dequeues

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].

Configured Subscription: a Subscription installed via a configuration interface which persists across reboots.

Dynamic Subscription: a Subscription negotiated between Subscriber and Publisher via create, establish, modify, and delete RPC signaling messages.

Event: an occurrence of something that may be of interest. (e.g., a configuration change, a fault, a change in status, crossing a threshold, status of a flow, or an external input to the system.)

Event Notification: a set of information intended for a Receiver indicating that one or more Event(s) have occurred. Details of the Event(s) may be included within.

Event Stream: a continuous, ordered set of Events grouped under an explicit criteria.

Notification: the communication of an occurrence, perhaps triggered by the occurrence of an Event.

Publisher: an entity responsible for streaming Event Notifications per the terms of a Subscription.

Receiver: a target to which a Publisher pushes Event Notifications. For Dynamic Subscriptions, the Receiver and Subscriber will often be the same entity.

Subscriber: an entity able to request and negotiate a contract for the receipt of Event Notifications from a Publisher

Subscription: a contract between a Subscriber and a Publisher stipulating which information the Receiver wishes to have pushed from the Publisher without the need for further solicitation.

3. Solution

Event subscription is defined in [rfc5277bis], YANG Datastore subscription is defined in [yang-push]. This section specifies transport mechanisms applicable to both.

3.1. Mechanisms for Subscription Establishment and Maintenance

There are three models for Subscription establishment and maintenance:

1. Dynamic Subscription: Here the Subscriber and Receiver are the same. A Subscription ends with a subscription-terminated notification, or by a loss of transport connectivity.

2. Configured Subscription: Receiver(s) are specified on Publisher in startup and running config. Subscription is not terminated except via an operations interface. (Subscriptions may be Suspended, with no Event Notifications sent however.)

3. Proxy Subscription: Subscriber and Receiver are different. Subscription ends when a Subscription End-time is reached, or the Publisher process is restarted. A key difference between this and configured subscriptions (#2) is that configuration requests are made to RPCs which might evaluate run-time conditions much like in (#1). Typically direct configuration via (#2) will not go through the same sort of capacity and validation checks seen in (#1).

The first two models are described in this section. The third is described in Appendix A. This third model will be moved into the body of this specification should the IETF community desire. In theory, all three models may be intermixed in a single deployment.

```
                          .---------------.
                          |   Publisher   |
                          '---------------'
                            ^   ^   |   ^
                            |   |   |   |
          .-----Restconf----'   |   |   '-----Restconf----.
          |               .-----'   '-HTTP-.              |
          V               |                V              |
   .-------------.  .------------.  .----------.  .------------.
   | Subscriber+ |  | Operations |  | Receiver |  | Subscriber |
   | Receiver    |  |  /Config   |  '----------'  '------------'
   '-------------'  '------------'    ^     ^               ^
          ^         (out of scope)    :     :               :
          :             ^             :     :...Model #3....:
       Model #1         :..Model #2...:      (out of scope)
```

Figure 1: Subscription Models

3.2. Dynamic YANG Subscription with RESTCONF control

Dynamic Subscriptions for both [rfc5277bis] and its [yang-push] augmentations are configured and managed via signaling messages transported over [restconf]. These interactions will be accomplished via a Restconf POST into RPCs located on the Publisher. HTTP responses codes will indicate the results of the interaction with the Publisher. An HTTP status code of 200 is the proper response to a successful <establish-subscription> RPC call. The successful <establish-subscription> will result in a HTTP message with returned subscription URI on a logically separate mechanism than was used for the original Restconf POST. This mechanism would be via a parallel TCP connection in the case of HTTP 1.x, or in the case of HTTP2 via a separate HTTP stream within the HTTP connection. When a being returned by the Publisher, failure will be indicated by error codes transported in payload, as well as the return of negotiation parameters.

Once established, streaming Event Notifications are then delivered via SSE for HTTP1.1 and via HTTP Data for HTTP2.

3.2.1. Call Flow for HTTP2

Requests to [yang-push] augmented RPCs are sent on one or more HTTP2 streams indicated by (a) in Figure 2. Event Notifications related to a single subscription are pushed on a unique logical channel (b). In the case below, a newly established subscription has its events pushed over HTTP2 stream (7).

```
   +------------+                               +------------+
   | Subscriber |                               | Publisher  |
   |HTTP2 Stream|                               |HTTP2 Stream|
   |  (a)  (b)  |                               |  (a)  (b)  |
   +------------+                               +------------+
       | Restconf POST (RPC:establish-subscription)   |
       |--------------------------------------------->|
       |                             HTTP 200 OK (URI)|
       |<---------------------------------------------|
       |   (7)HTTP POST (URI)                             (7)
       |    |--------------------------------------------->|
       |    |                                  HTTP 200 OK|
       |    |<---------------------------------------------|
       |    |                      HTTP Data (event-notif)|
       |    |<---------------------------------------------|
       | Restconf POST (RPC:modify-subscription)      |    |
       |--------------------------------------------->|    |
       |    |                            HTTP 200 OK|    |
       |<---------------------------------------------|    |
       |    |            HTTP Data (subscription-modified)|
       |    |<---------------------------------------------|
       |    |                      HTTP Data (event-notif)|
       |    |<---------------------------------------------|
       | Restconf POST (RPC:delete-subscription)      |    |
       |--------------------------------------------->|    |
       |    |                            HTTP 200 OK|    |
       |<---------------------------------------------|    |
       |    |                 HTTP Headers (end of stream)|
       |   (/7)<------------------------------------------(/7)
       |
```

                         Figure 2: Dynamic with HTTP2

3.2.2. Call flow for HTTP1.1

Requests to [yang-push] RPCs are sent on the TCP connection indicated by (a). Event Notifications are pushed on a separate connection (b). This connection (b) will be used for all Event Notifications across all subscriptions.

```
   +--------------+                             +--------------+
   |  Subscriber  |                             |   Publisher  |
   |TCP connection|                             |TCP connection|
   |  (a)  (b)    |                             |    (a)  (b)  |
   +--------------+                             +--------------+
       | Restconf POST (RPC:establish-subscription)   |
       |--------------------------------------------->|
       |                             HTTP 200 OK (URI)|
       |<---------------------------------------------|
       |    |HTTP GET (URI)                                |
       |    |--------------------------------------------->|
       |    |                                   HTTP 200 OK|
       |    |<---------------------------------------------|
       |    |                             SSE (event-notif)|
       |    |<---------------------------------------------|
       | Restconf POST (RPC:modify-subscription)      |    |
       |--------------------------------------------->|    |
       |    |                              HTTP 200 OK|    |
       |<---------------------------------------------|    |
       |    |                   SSE (subscription-modified)|
       |    |<---------------------------------------------|
       |    |                             SSE (event-notif)|
       |    |<---------------------------------------------|
       | Restconf POST (RPC:delete-subscription)      |    |
       |--------------------------------------------->|    |
       |    |                              HTTP 200 OK|    |
       |<---------------------------------------------|    |
       |    |                                              |
       |    |
```

Figure 3: Dynamic with HTTP1.1

3.2.3. Configured Subscription over HTTP2

With a Configured Subscription, all information needed to establish a secure relationship with that Receiver is available on the Publisher. With this information, the Publisher will establish a secure transport connection with the Receiver and then begin pushing the Event Notifications to the Receiver. Since Restconf might not exist on the Receiver, it is not desirable to require that such Event Notifications be pushed with any dependency on Restconf. Nor is there value which Restconf provides on top of HTTP. Therefore in place of Restconf, a TLS secured HTTP2 Client connection must be established with an HTTP2 Server located on the Receiver. Event Notifications will then be sent as part of an extended HTTP POST to the Receiver.

POST messages will be addressed to HTTP augmentation code on the Receiver capable of accepting and responding to Event Notifications. The first POST message must be a subscription-started notification. Push update notifications must not be sent until the receipt of an HTTP 200 OK for this initial notification. The 200 OK will indicate that the Receiver is ready for Event Notifications. At this point a Subscription must be allocated its own HTTP2 stream. Figure 4 depicts this message flow.

```
+------------+                                 +------------+
|  Receiver  |                                 | Publisher  |
|HTTP2 Stream|                                 |HTTP2 Stream|
|  (a)  (b)  |                                 |  (a)  (b)  |
+------------+                                 +------------+
    |    HTTP Post Headers, Data (sub-start, SubID)|
    |<-----------------------------------------------|
    | HTTP 200 OK                                    |
    |----------------------------------------------->|
    |    |         HTTP Post Headers, Data (event-notif)|
    |    |<-----------------------------------------------|
    |    |                        HTTP Data (event-notif)|
    |    |<-----------------------------------------------|
    |    |                      HTTP Data (sub-terminate)|
    |    |<-----------------------------------------------|
    |    |HTTP 200 OK                                     |
    |    |----------------------------------------------->|
```

Figure 4: Configured over HTTP2

As the HTTP2 transport is available to the Receiver, the Publisher should:

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

- o take a subscription-dependency if it has been provided and map the HTTP2 stream for the parent subscription into the HTTP2 stream dependency.

3.3. Subscription Multiplexing

It is possible that updates might be delivered in a different sequence than generated. Reasons for this might include (but are not limited to):

- o replay of pushed updates

- temporary loss of transport connectivity, with update buffering and different dequeuing priorities per Subscription

- population, marshalling and bundling of independent Subscription Updates, and

Therefore each Event Notification will include a millisecond level timestamp to ensure that a Receiver understands the time when a that update was generated. Use of this timestamp can give an indication of the state of objects at a Publisher when state-entangled information is received across different subscriptions. The use of the latest Event Notification timestamp for a particular object update can introduce errors. So when state-entangled updates have inconsistent object values and temporally close timestamps, a Receiver might consider performing a GET to validate the current state of a Publisher.

3.4. Encoded Subscription and Event Notification Examples

Transported updates will contain context data for one or more Event Notifications. Each transported Event Notification will contain several parameters:

3.4.1. Restconf Subscription and Events over HTTP1.1

Subscribers can dynamically learn whether a RESTCONF server supports various types of Event or Yang datastore subscription capabilities. This is done by issuing an HTTP request OPTIONS, HEAD, or GET on the stream. Some examples building upon the Call flow for HTTP1.1 from Section 3.2.2 are:

```
GET /restconf/data/ietf-restconf-monitoring:restconf-state/
         streams/stream=yang-push HTTP/1.1
Host: example.com
Accept: application/yang.data+xml
```

If the server supports it, it may respond

```
HTTP/1.1 200 OK
Content-Type: application/yang.api+xml
<stream xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf-monitoring">
            <name>yang-push</name>
            <description>Yang push stream</description>
            <access>
               <encoding>xml</encoding>
               <location>https://example.com/streams/yang-push-xml
               </location>
            </access>
            <access>
               <encoding>json</encoding>
               <location>https://example.com/streams/yang-push-json
               </location>
            </access>
         </stream>
```

If the server does not support any form of subscription, it may respond

```
HTTP/1.1 404 Not Found
Date: Mon, 25 Apr 2012 11:10:30 GMT
Server: example-server
```

Subscribers can determine the URL to receive updates by sending an HTTP GET as a request for the "location" leaf with the stream list entry. The stream to use for may be selected from the Event Stream list provided in the capabilities exchange. Note that different encodings are supporting using different Event Stream locations. For example, the Subscriber might send the following request:

```
GET /restconf/data/ietf-restconf-monitoring:restconf-state/
         streams/stream=yang-push/access=xml/location HTTP/1.1
Host: example.com
Accept: application/yang.data+xml
```

The Publisher might send the following response:

```
HTTP/1.1 200 OK
Content-Type: application/yang.api+xml
   <location
        xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf-monitoring">
        https://example.com/streams/yang-push-xml
   </location>
```

To subscribe and start receiving updates, the subscriber can then send an HTTP GET request for the URL returned by the Publisher in the request above. The accept header must be "text/event-stream". The

Publisher uses the Server Sent Events [W3C-20150203] transport strategy to push filtered Event Notifications from the Event stream.

The Publisher MUST support individual parameters within the POST request body for all the parameters of a subscription. The only exception is the encoding, which is embedded in the URI. An example of this is:

```
// subtree filter = /foo
// periodic updates, every 5 seconds
POST /restconf/operations/ietf-event-notifications:
     establish-subscription HTTP/1.1
      Host: example.com
      Content-Type: application/yang-data+json

      {
        "ietf-event-notifications:input" : {
          ?stream?: ?push-data"
          ?period" : 5,
          "xpath-filter" : ?/ex:foo[starts-with(?bar?.?some']"
        }
      }
```

Should the publisher not support the requested subscription, it may reply:

```
HTTP/1.1 501 Not Implemented
Date: Mon, 23 Apr 2012 17:11:00 GMT
Server: example-server
Content-Type: application/yang.errors+xml
    <errors xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf">
       <error>
           <error-type>application</error-type>
           <error-tag>operation-not-supported</error-tag>
           <error-severity>error</error-severity>
           <error-message>Xpath filters not supported</error-message>
           <error-info>
               <supported-subscription xmlns="urn:ietf:params:xml:ns:
                   netconf:datastore-push:1.0">
                   <subtree-filter/>
               </supported-subscription>
           </error-info>
       </error>
     </errors>
```

with an equivalent JSON encoding representation of:

```
HTTP/1.1 501 Not Implemented
Date: Mon, 23 Apr 2012 17:11:00 GMT
Server: example-server
Content-Type: application/yang.errors+json
      {
        "ietf-restconf:errors": {
          "error": {
            "error-type": "protocol",
            "error-tag": "operation-not-supported",
            "error-message": "Xpath filters not supported."
            "error-info": {
               "datastore-push:supported-subscription": {
                     "subtree-filter": [null]
                 }
            }
          }
        }
      }
```

The following is an example of a pushed Event Notification data for the Subscription above. It contains a subtree with root foo that contains a leaf called bar:

XML encoding representation:

```
  <?xml version="1.0" encoding="UTF-8"?>
  <notification xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf">
     <subscription-id xmlns="urn:ietf:params:xml:ns:restconf:
         datastore-push:1.0">
           my-sub
     </subscription-id>
     <eventTime>2015-03-09T19:14:56.23Z</eventTime>
     <datastore-contents xmlns="urn:ietf:params:xml:ns:restconf:
        datastore-push:1.0">
        <foo xmlns="http://example.com/yang-push/1.0">
          <bar>some_string</bar>
        </foo>
     </datastore-contents>
  </notification>
```

Or with the equivalent YANG over JSON encoding representation as defined in [RFC7951]:

```
{
  "ietf-restconf:notification": {
    "datastore-push:subscription-id": "my-sub",
    "eventTime": "2015-03-09T19:14:56.23Z",
    "datastore-push:datastore-contents": {
      "example-mod:foo": { "bar": "some_string" }
    }
  }
}
```

To modify a Subscription, the subscriber issues another POST request on the provided URI using the same subscription-id as in the original request. For example, to modify the update period to 10 seconds, the subscriber may send:

```
POST /restconf/operations/ietf-event-notifications:
      modify-subscription HTTP/1.1
      Host: example.com
      Content-Type: application/yang-data+json

      {
        "ietf-event-notifications:input" : {
          ?subscription-id?: 100,
          ?period" : 10
        }
      }
```

To delete a Subscription, the Subscriber issues a DELETE request on the provided URI using the same subscription-id as in the original request

3.4.2. Event Notification over HTTP2

The basic encoding will look as below. It will consists of a JSON representation wrapped in an HTTP2 header.

```
HyperText Transfer Protocol 2
      Stream: HEADERS, Stream ID: 5
      Header: :method: POST
      Stream: HEADERS, Stream ID: 5

{
  "ietf-yangpush:notification": {
    "datastore-push:subscription-id": "my-sub",
    "eventTime": "2015-03-09T19:14:56.23Z",
    "datastore-push:datastore-contents": {
      "foo": { "bar": "some_string" }
    }
  }
}
```

3.5. Stream Discovery

Relevant for Dynamic Subscriptions, this will be accomplished as specified in [restconf] section 6.2. The namespace chosen will be the same as how stream names are acquired for NETCONF, and so that backwards compatibility can be maintained without replicating this information.

As per [restconf] section 6.3, RESTCONF clients can determine the URL for the subscription resource (to receive notifications) by sending an HTTP GET request for the "location" leaf with the stream list entry.

4. Security Considerations

Subscriptions could be used to intentionally or accidentally overload the resources of a Publisher. For this reason, it is important that the Publisher has the ability to prioritize the establishment and push of Event Notifications where there might be resource exhaust potential. In addition, a server needs to be able to suspend existing Subscriptions when needed. When this occurs, the subscription status must be updated accordingly and the Receivers notified.

A Subscription could be used to attempt retrieve information for which a Receiver has no authorized access. Therefore it is important that data pushed via a Subscription is authorized equivalently with regular data retrieval operations. Data being pushed to a Receiver needs therefore to be filtered accordingly, just like if the data were being retrieved on-demand. The Netconf Authorization Control Model [RFC6536] applies even though the transport is not NETCONF.

One or more Publishers of Configured Subscriptions could be used to overwhelm a Receiver which doesn't even support Subscriptions. There are two protections here. First Event Notifications for Configured Subscriptions MUST only be transmittable over Encrypted transports. Clients which do not want pushed Event Notifications need only terminate or refuse any transport sessions from the Publisher. Second, the HTTP transport augmentation on the Receiver must send an HTTP 200 OK to a subscription started notification before the Publisher starts streaming any events.

One or more Publishers could overwhelm a Receiver which is unable to control or handle the volume of Event Notifications received. In deployments where this might be a concern, HTTP2 transport such as HTTP2) should be selected.

5. Acknowledgments

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

6. References

6.1. Normative References

[restconf]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", March 2016, <https://datatracker.ietf.org/doc/draft-ietf-netconf-restconf/>.

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

[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, <http://www.rfc-editor.org/info/rfc6520>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <http://www.rfc-editor.org/info/rfc6536>.

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

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

[RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <http://www.rfc-editor.org/info/rfc7951>.

6.2. Informative References

[call-home]
Watsen, K., "NETCONF Call Home and RESTCONF Call Home", December 2015, <https://tools.ietf.org/html/draft-ietf-netconf-call-home-17>.

[rfc5277bis]
Gonzalez Prieto, A., Clemm, A., Voit, E., Prasad Tripathy, A., and E. Nilsen-Nygaard, "NETCONF Event Notifications", September 2016, <https://datatracker.ietf.org/doc/draft-ietf-netconf-rfc5277bis/>.

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

[yang-push]
Clemm, A., Voit, E., Gonzalez Prieto, A., Prasad Tripathy, A., and E. Nilsen-Nygaard, "Subscribing to YANG datastore push updates", June 2016, <https://datatracker.ietf.org/doc/draft-ietf-netconf-yang-push/>.

Appendix A. Proxy YANG Subscription when the Subscriber and Receiver are different

The properties of Dynamic and Configured Subscriptions can be combined to enable deployment models where the Subscriber and Receiver are different. Such separation can be useful with some combination of:

- o An operator does not want the subscription to be dependent on the maintenance of transport level keep-alives. (Transport independence provides different scalability characteristics.)
- o There is not a transport session binding, and a transient Subscription needs to survive in an environment where there is unreliable connectivity with the Receiver and/or Subscriber.
- o An operator wants the Publisher to include highly restrictive capacity management and Subscription security mechanisms outside of domain of existing operational or programmatic interfaces.

To build a Proxy Subscription, first the necessary information must be signaled as part of the <establish-subscription>. Using this set of Subscriber provided information; the same process described within section 3 will be followed. There is one exception. Only when an HTTP status code of 200 comes back from the receiver, will it inform the Subscriber of Subscription establishment success via its Restconf connection.

After a successful establishment, if the Subscriber wishes to track the state of Receiver subscriptions, it may choose to place a separate on-change Subscription into the "Subscriptions" subtree of the YANG Datastore on the Publisher.

Appendix B. End-to-End Deployment Guidance

Several technologies are expected to be seen within a deployment to achieve security and ease-of-use requirements. These are not necessary for an implementation of this specification, but will be useful to consider when considering the operational context.

B.1. Call Home

Pub/Sub implementations should have the ability to transparently incorporate [call-home] so that secure TLS connections can originate from the desired device.

B.2. TLS Heartbeat

HTTP sessions might not quickly allow a Subscriber to recognize when the communication path has been lost from the Publisher. To recognize this, it is possible for a Receiver to establish a TLS heartbeat [RFC6520]. In the case where a TLS heartbeat is included, it should be sent just from Receiver to Publisher. Loss of the heartbeat should result in any Subscription related TCP sessions between those endpoints being torn down. The subscription can then attempt to re-establish.

Appendix C. Issues being worked and resolved

(To be removed by RFC editor prior to publication)

C.1. Unresolved Issues

RT3 - Do we include 3rd party signaled subscriptions within models that need to be supported generically, or for a particular type of transport.

RT10 - Right now the examples show a YANG timestamp at the hundredths of a second level. But the yang-push draft is at seconds. And the requirements show at least milliseconds (if not more).

C.2. Agreement in principal

RT4 - Need to add into document examples of 5277bis Event streams. Document only includes yang-push examples at this point.

RT6 - We need to define encodings of rfc5277bis notifications.

C.3. Resolved Issues

RT1 - Integration specifics for Restconf capability discovery on different types of Streams

RT2 - In what way to we position Event notifications model in this document vs. current solution in Restconf.

RT5 - Doesn't make sense to use Restconf for Configured subscriptions. HTTP will be used.

RT7 - HTTP native option doesn't currently use SSE. But we should evaluate moving to that as possible. It will make development integration easier and more consistent.

RT8 - Once SSE starts, there will be no more Restconf interpretation of further signaling upon the connection. It is unclear how this can be made to work with modify and delete subscription. If it cannot, a method of sending events without SSE will be needed, although this would diverge from the existing Restconf mechanisms

RT9 - For static subscriptions, perhaps we can use Restconf call home to originate an SSE connection. This assume RT8 & RT2 can be resolved with SSE.

Appendix D. Changes between revisions

(To be removed by RFC editor prior to publication)

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

Alexander Clemm
Cisco Systems

Email: alex@clemm.org

Alberto Gonzalez Prieto
Cisco Systems

Email: albertgo@cisco.com

Ambika Prasad Tripathy
Cisco Systems

Email: ambtripa@cisco.com

Einar Nilsen-Nygaard
Cisco Systems

Email: einarnn@cisco.com

Andy Bierman
YumaWorks

Email: andy@yumaworks.com