Requirements for Subscription to YANG Datastores
draft-voit-i2rs-pub-sub-requirements-00

Abstract

   This document provides requirements for a service that allows client
   applications to subscribe to updates of a YANG datastore. Based on
   criteria negotiated as part of a subscription, updates will be pushed
   to targeted recipients. Such a capability eliminates the need for
   periodic polling of YANG datastores by applications and fills a
   functional gap in existing YANG transports (i.e. Netconf and
   Restconf). Such a service can be summarized as a "pub/sub" service
   for YANG datastore updates. Beyond a set of basic requirements for
   the service, various refinements are addressed. These refinements
   include: periodicity of object updates, filtering out of objects
   underneath a requested a subtree, and delivery QoS guarantees.

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 http://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 June 13, 2015.

Copyright Notice

Table of Contents

1. Introduction

   YANG has gained acceptance as the data definition language of choice
   for control and management related information. Applications that
   interact with YANG datastores are extending beyond traditional
   configuration of network elements. In many cases these applications
   are aimed at service-assurance, which involves monitoring of
   operational data and state. The existing YANG technology ecosystem
   is proving insufficient for those applications due to:

   o a reliance on RPC-style interactions where data is configured or
      fetched on-demand by applications.

   o limited support for pushed notification of changes. (Currently
      notifications are for configuration data only. And even then,
      subscription mechanisms for such notifications are undefined.)

Put simply, periodic fetching of data is not an adequate solution for applications requiring frequent or prompt updates of remote object state. Trying to impose a polling based solution to this problem imposes load on networks, devices, and applications. Additionally, polling solutions are brittle in the face of communication glitches, and they have limitations in their ability to synchronize and calibrate retrieval intervals across a network.

I2RS WG documents have expressed a need for more robust YANG object subscriptions. Similar discussions are underway in NETMOD and NETCONF. With the support of standards bodies such as OMG (DDS) , XMPP.org standard, generic Pub/Sub mechanisms to communicate data updates have been defined and proven themselves in a wide variety of deployments.

It is time to incorporate such generic object subscription mechanisms between Network Elements, and allow these mechanisms to be applied in the context of data that is conceptually contained in YANG datastores. With such mechanisms, local Network Element based applications can have access to a set of consistent network information driven via push from peer Network Elements which host authoritative information.

There are some decent IETF starting points and contexts for these mechanisms. For example Netconf Event Notifications [RFC5277] provides a useful tool for an end-to-end solution. However RFC5277 does not follow the Pub/Sub paradigm and predates YANG. [RFC6470] defines configuration change notifications, but is applicable for configuration information only.

Because of this, the authors have put forward this requirements document as well as [datastore-push]. We are hoping these could provide a context upon which to create new solution.

2. Business Drivers

We need to move beyond the paradigm of periodic retrieval of data. We need to move more towards robust set of object state maintenance available between network elements. For decades, information delivery between peering network elements has been accomplished by dedicated, customized networking protocols. For example:

o Routing protocols have used broadcast flooding to ensure timely replication of link and node state.

o Auto-negotiation protocols have shared capabilities so that devices could settle on joint configuration of link endpoints.

With the growth of SDN, imperative policy distribution, and YANG's
ascent as a dominant programmatic interface to network elements,
custom protocol development is no longer sufficient. What is needed
is a more generic information distribution mechanism that is able to
deliver objects between peering network elements.

These generic information distribution mechanisms will not replace
existing protocols. Instead they will supplement these protocols.
However in the absence of a customized networking protocol, an
alternative mechanism is needed to alert a data supplier (Publisher)
that a data consumer (Subscriber) wants rapid delivery when specific
data object(s) change.

At the same time, SNMP and MIBs are still widely deployed and the de-
facto choice for many monitoring solutions. Those solutions do not
require support for configuration transactions and the need to
validate and maintain configuration consistency, hence there is less
pressure to abandon SNMP and MIBs. Arguably the biggest shortcoming
of SNMP for those applications concerns the need to rely on periodic
polling, because it introduces additional load on the network and
devices, is brittle in case polling cycles are missed, and is hard to
synchronize and calibrate across a network, making data obtained from
multiple devices less comparable. If applications need to apply
those same interaction patterns for YANG datastores, similar issues
can be expected. Migration to YANG datastores by applications that
do not have to worry about transactional integrity becomes a lot more
compelling if those issues are addressed.

2.1. Pub/Sub in I2RS

Various I2RS documents highlight the need to provide Pub/Sub
capabilities between network elements. From [i2rs-arch], there are
references throughout the document beginning in section 6.2. Some
specific examples include:

o section 7.6 provides high level pub/sub (notification) guidance

o section 6.4.2 identifies "subscribing to an information stream of
   route changes receiving notifications about peers coming up or
   going down"

o section 6.3 notes that when local config preempts I2RS, external
   notification might be necessary

In addition [i2rs-usecase]has relevant requirements. A small subset
includes:

o L-Data-REQ-12: The I2RS interface should support user
   subscriptions to data with the following parameters: push of data
   synchronously or asynchronously via registered subscriptions...

o L-DATA-REQ-07: The I2RS interface (protocol and IMs) should allow
   a subscribe to select portions of the data model.

o PI-REQ01: monitor the available routes installed in the RIB of
   each forwarding device, including near real time notification of
   route installation and removal.

o BGP-REQ10: I2RS client SHOULD be able instruct the I2RS agent(s)
   to notify the I2RS client when the BGP processes on an associated
   routing system observe a route change to a specific set of IP
   Prefixes and associated prefixes....The I2RS agent should be able
   to notify the client via publish or subscribe mechanism.

o IGP-REQ-07: The I2RS interface (protocol and IMs) should support a
   mechanism where the I2RS Clients can subscribe to the I2RS Agent's
   notification of critical node IGP events.

o MPLS-LDP-REQ-03: The I2RS Agent notifications should allow an I2RS
   client to subscribe to a stream of state changes regarding the LDP
   sessions or LDP LSPs from the I2RS Agent.

o L-Data-REQ-01: I2rs must be able to collect large data set from
   the network with high frequency and resolution with minimal impact
   to the device's CPU and memory.

There are additional individual drafts such as [i2rs-pubsub-security]
documenting the Pub/Sub needs for: time delivery sensitivity, support
for multiple transport protocols, secure/authorized communications,
and support for a range specification of subscribed data delivery
content. So the list above should not be considered exhaustive.

2.2. Pub/Sub variants on Network Elements

Looking at history, there are many examples of switching and routiing
protocols which have done explicit or implicit pub/sub in the past.
In addition, new policy notification mechanisms which operate on
Switches and Routers are being specified now. A very small subset of
these includes:

o Routing Adjacencies in MPLS VPNs [RFC6513]

o OSPF Route Flooding [RFC2328]

o Multicast topology establishment protocols (IGMP, PIM, etc.)

   o Audio-Video Bridging streams needing guaranteed latency
      [AVB-latency] (802.1Q-2011 Clause 35)

   o Secure Automation and Continuous Monitoring (SACM)
      [sacm-requirements]

   o "Peer Mount" subscriptions for configuration verification between
      peers[draft-voit-netmod]

   Worthy of note in the list above is the wide variety of broadcast,
   multicast, and unicast transports used. In addition some transports
   are at L3, and some at L2. Therefore if we are going to attempt a
   generic Pub/Sub mechanism, it will need to be structured so that it
   may support alternative transports. Looking at the nearer term based
   on current I2RS requirements, NETCONF should be our transport
   starting point as it supports connection oriented/Unicast
   communication. But we need to be prepared to decouple where viable
   to support Multicast and Broadcast distribution as well.

2.3. Existing Generalized Pub/Sub Implementations

   TIBCO, RSS, CORBA, and other technologies all show precursor Pub/Sub
   technologies. However there are new needs described in Section 4
   below which these technologies do not serve. We need a technology.

   There are at least two widely deployed generalized pub/sub
   implementations which come close to current needs: XMPP[XEP-0060] and
   DDS[OMG-DDS]. Both serve as proof-points that a highly scalable
   distributed datastore implementation connecting millions of edge
   devices is possible.

   Because of these proof points, we can be comfortable that the
   underlying technologies can enable reusable generalized YANG object
   distribution. Analysis will need to fully dimension the speed and
   scale of such object distribution for various subtree sizes and
   transport types.

3. Terminology

   A Subscriber makes requests for set(s) of YANG object data. The
   Subscriber is the owner of the Subscription.

   A Publisher is responsible for distributing subscribed YANG object
   data per the terms of a Subscription. In general, a Publisher is the
   owner of the YANG datastore that is subjected to the Subscription.

   A Receiver is the target where a Publisher pushes updates. In
   general, the Receiver and Subscriber will be the same entity. A

Subscription Service provides Subscriptions to Subscribers of YANG data.

A Subscription Service interacts with the Publisher of the YANG data as needed to provide the data per the terms of the Subscription.

A Subscription Request for one or more YANG subtrees made by the Subscriber of a Publisher and targeted to a Receiver. A Subscription MAY include constraints which dictates how often or under what conditions YANG subtree updates might be sent.

A Subscription is a contract between a Subscription Service and a Subscriber that stipulates the data to be pushed and the associated terms.

A YANG datastore is a conceptual datastore that contains hierarchical data defined in YANG data models. It is what is referred in existing RFCs as "Netconf datastore". However, as the same datastore is no longer tied to Netconf as a specific transport, the term "YANG datastore" is deemed more appropriate.

4. Requirements

Many of the requirements within this section have been morphed from OMG's DDS and XMPP.org's requirements specifications.

4.1. Assumptions for Subscriber Behavior

This document provides requirements for the Subscription Service. It does not define all the requirements for the Subscriber/Receiver. However In order to frame the desired behavior of the Subscription Service, it is important to specify key input constraints.

A Subscriber MUST be able to infer when a Subscription Service is no longer active and when no more updates are being sent.

A Subscriber SHOULD avoid attempting to establish multiple Subscriptions pertaining to the same information, i.e. referring to the same datastore YANG subtrees.

A Subscriber MAY provide QoS criteria to the Subscription Service such that if the Subscription Service is unable to meet those criteria, the Subscription should not be established.

When a Subscriber needs to restart, it is acceptable for the Subscriber to have to resubscribe. There is no requirement for the life span of the Subscription to extend beyond the life span of the Subscriber.

4.2. Subscription Service Requirements

4.2.1. General

A Subscription Service MUST support the ability to create and to
terminate a Subscription.

A Subscription Service MUST be able to support and independently
track one or more Subscription Requests by the same Subscriber.

A Subscription Service MUST be able to support an add/change/delete
of one or more YANG subtrees as part of the same Subscription
Request.

A Subscription Service MUST be able include a flag whether updates
pertain only to operational data, to configuration data, or both.

A Subscription may include filters as defined within a Subscription
Request, the Subscription Service MUST publish only data nodes that
meet the filter criteria.

A Subscription Service MUST support the ability to subscribe to
periodic updates. The subscription period SHOULD be configurable as
part of the request.

A Subscription Service SHOULD support the ability to subscribe to
updates "on-change", i.e. whenever values of subscribed data objects
change.

For "on-change" updates, the Subscription Service MUST support a
dampening period that needs to pass before the first or subsequent
"on-change" updates are sent. The dampening period SHOULD be
configurable as part of the subscription request.

A Subscription Service MUST allow Subscriptions to be monitored.
Specifically, a Subscription Service MUST at a minimum maintain
information about which Subscriptions are being serviced, the terms
of those subscriptions (e.g. what data is being subscribed,
associated filters, update policy - on change, periodic), and the
overall status of the Subscription - e.g. active or suspended.

A Subscription Service SHOULD be able to interpret Subscription
Requests QoS Policy requests, and only establish a Subscription if it
is possible to meet the QoS those QoS Policy requests.

A Subscription Service MUST support terminating of a Subscription
when requested by the Subscriber.

A Subscription Service SHOULD support the ability to suspend and to
resume a Subscription on request of a client.

A Subscription Service MAY at its discretion revoke or suspend an
existing subscription. Reasons MAY include transitory resource
limitation, credential expiry, failure to reconfirm a subscription,
loss of connectivity with the Receiver, operator CLI, and/or others.
When this occurs, the Subscription Service MUST notify the Subscriber
and update subscription status.

4.2.2. Negotiation

A Subscription Service MUST be able to negotiate the following terms
of a Subscription:

o The interval, for periodic publication policy

o The dampening period, for on-change update policy

o The policy: i.e. whether updates are on-change of periodic

o Any filters associated with a subtree subscription

A Subscription Service SHOULD be able to negotiate QoS criteria for a
Subscription. Examples of QoS criteria MAY include reliability of
the Subscription Service, reaction time between a monitored YANG
subtree/object change and a corresponding notification push, and the
Subscription Service's ability to support certain levels of object
liveliness.

In cases where a Subscription Request cannot be fulfilled, the
Subscription Service MUST include in its decline a set of criteria
that would have been acceptable when the Subscription Request was
made. For example, if periodic updates were requested with too short
update intervals for the specified data set, the minimum acceptable
interval period SHOULD be included. If on-change updates were
requested with a dampening period, the minimum acceptable dampening
period SHOULD be included, or an indication whether only periodic
updates are supported along with the minimum acceptable interval
period for the data set being subscribed to.

4.2.3. Update Distribution

For "on-change" updates, the Subscription Service MUST only send
deltas to the object data for which a change occurred. [Otherwise
the subscriber will not know what has actually undergone change.]
The updates for each object needs to include an indication whether it
was removed, added, or changed.

When a Subscription Service is not able to send updates per its
subscription contract, the Subscription MUST notify subscribers and
put the subscription into a state of indicating the Subscription was
suspended by the service. When able to resume service, subscribers
need to be notified as well. If unable to resume service, the
Subscription Service MAY terminate the subscription and notify
Subscribers accordingly.

When a Subscription with "on-change" updates is suspended and then
resumed, the first update SHOULD include updates of any changes that
occurred while the Subscription was suspended, with the current
value. The Subscription Service MUST provide a clear indication when
this capability is not supported (because in this case a client
application may have to synchronize state separately).

A Subscription Service MUST send an indication to the Subscriber when
a Subscription undergoes a state change, i.e. when it is started,
suspended, resumed, or terminated.

A Subscription Service MAY, as an option, support a persistence/
replay capability.

[4.2.4](#). Transport

A Subscription Service SHOULD support different transports.

A Subscription Service SHOULD support different encodings of payload.

It MUST be possible for Receivers to associate the update with a
specific Subscription.

In the case of connection-oriented transport, when a transport
connection drops, the associated Subscription SHOULD be terminated.
It is up the Subscriber to request a new Subscription.

[4.2.5](#). Security Requirements

As part of the Subscription establishment, there must be mutual
authentication between the Subscriber and the Subscription Service.

When there are multiple Subscribers, it should be possible to provide
cryptographic authentication in such a way that no Subscriber can
pose as the orginal Subscription Service.

Versioning MUST be supported.

A Subscription could be used to retrieve data in subtrees that a
client has not authorized access to. Therefore it is important that

data pushed based on Subscriptions is authorized in the same way that regular data retrieval operations are. Data being pushed to a client needs therefore to be filtered accordingly, just like if the data were being retrieved on-demand. For Unicast transports, the Netconf Authorization Control Model applies.

Subscription requests, including requests to create, terminate, suspend, and resume Subscriptions MUST be properly authorized.

A Subscription Service MUST filter Subscriptions to suppress object updates where the Receiver has no read authorization. The Subscription Service MUST integrate with NACM (Netconf Access Control Model) or other transport specific access control mechanisms as applicable.

When the Subscriber and Receiver are different, the Receiver MUST be able to terminate any Subscription to it where objects are being delivered over a Unicast transport.

A Subscription Service SHOULD decline a Subscription Request if it would deplete its resources. It is preferable to decline a Subscription when originally requested, rather than having to terminate it prematurely later.

5. Acknowledgements

We wish to acknowledge the helpful contributions, comments, and suggestions that were received from Ambika Tripathy and Prabhakara Yellai as well as the helpfulness of related end-to-end system context from [i2rs-pubsub-security] from Nancy Cam Winget, Ken Beck, and David McGrew.

6. References

6.1. Normative References

[RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, April 1998.

[RFC5277] Chisholm, S. and H. Trevino, "NETCONF Event Notifications", RFC 5277, July 2008.

[RFC6470] Bierman, A., "Network Configuration Protocol (NETCONF) Base Notifications", RFC 6470, February 2012.

[RFC6513] Rosen, E. and R. Aggarwal, "Multicast in MPLS/BGP IP VPNs", RFC 6513, February 2012.

6.2. Informative References

   [AVB-latency]
               Jeffree, T., "802.1Qav - Forwarding and Queuing
               Enhancements for Time-Sensitive Streams", December 2009,
               <http://www.ieee802.org/1/pages/802.1av.html>.

   [OMG-DDS]   "Data Distribution Service for Real-time Systems, version
               1.2", January 2007, <http://www.omg.org/spec/DDS/1.2/>.

   [XEP-0060]
               Millard, P., "XEP-0060: Publish-Subscribe", July 2010,
               <XEP-0060: Publish-Subscribe>.

   [datastore-push]
               Clemm, A., Gonzalez Prieto, A., and E. Voit, "Subscribing
               to datastore push updates", October 2014,
               <https://tools.ietf.org/html/draft-netmod-clemm-datastore-
               push-00>.

   [draft-voit-netmod]
               Voit, E., "Requirements for Peer Mounting of YANG subtrees
               from Remote Datastores", October 2014,
               <https://tools.ietf.org/html/draft-voit-netmod-peer-mount-
               requirements-01>.

   [i2rs-arch]
               Atlas, A., "An Architecture for the Interface to the
               Routing System", December 2014,
               <https://tools.ietf.org/html/draft-ietf-i2rs-architecture-
               06>.

   [i2rs-pubsub-security]
               Beck, K., Cam Winget, N., and D. McGrew, "Using the
               Publish-Subscribe Model in the Interface to the Routing
               System", July 2013, <https://tools.ietf.org/html/draft-
               camwinget-i2rs-pubsub-sec-00>.

   [i2rs-usecase]
               Hares, S. and M. Chen, "Summary of I2RS Use Case
               Requirements", October 2014,
               <https://datatracker.ietf.org/doc/draft-ietf-i2rs-usecase-
               reqs-summary/>.

   [sacm-requirements]
              Cam Winget, N., "Secure Automation and Continuous
              Monitoring (SACM) Requirements", October 2014,
              <https://tools.ietf.org/html/draft-ietf-sacm-requirements-
              02>.

Authors' Addresses

   Eric Voit
   Cisco Systems

   Email: evoit@cisco.com

   Alex Clemm
   Cisco Systems

   Email: alex@cisco.com

   Alberto Gonzalez Prieto
   Cisco Systems

   Email: albertgo@cisco.com