YANG Push and ConfD

tail-f a Cisco company

Table of Contents

YANG Push and ConfD

1. Introduction

Network operators need to be able to collect and analyze statistical, performance, and status information from the devices in their network in order to monitor the health of their network, to gather information for billing purposes, and to be aware of changes that may occur in their networks. With the increased use of automation and the emergence of concepts such as Intent Based Networking (IBN), systems need to collect more operational data than ever before.

This collection of information has been traditionally done from central locations in the network by requesting the information from the devices i.e. a pull model accomplished by polling the devices. One way this has been done by using SNMP to periodically request the different types of information from an SNMP agent running on the device. The requests can be made at various times and frequencies, depending on the when the information needs to be collected, but the collection always is initiated by a centralized management station.

Using SNMP to collect operational data such as statistical, performance, and status information does introduce some problems. SNMP uses UDP as its transport. As a result, requests can be lost or delayed as they proceed through the network, particularly if the network is experiencing high load or congestion. The data pulled in may be incomplete or may not be accurate in terms of when the information should be collected. Another weakness of using SNMP, and any other technique which requires the information to be repeatedly requested or polled, is that often the same request is sent each time and must be parsed repeatedly by the agent on the network device. This consumes computing resources which may be scarce on the device. These problems may also be exacerbated if there is more than one collector in the network, perhaps requesting information for different purposes.

NETCONF and is also susceptible to some of these problems when used to collect statistical, performance and status information. Although a more reliable transport is used, namely TCP over SSH, this will introduce additional network overhead. Additionally, the problem of polling is not avoided, since a NETCONF <get> RPC must be sent each time data is needed and the NETCONF server on the device must parse and process each request.

The NETCONF Working Group within the IETF recognized that a more efficient way of gathering information needed to be devised. In particular, having a subscription mechanism that allows a collector to specify the information it wants to collect and the frequency at which the information should be delivered. This allows the network device to push the data rather than have the collector pull the data. After much work, the NETCONF Working Group has delivered a set of RFCs which are collectively referred to as "YANG Push".

This application will describe the parts of YANG Push with a focus on the implementation as it currently exists in ConfD 7.5 (January 2021).

2. YANG Push Overview

This section provides an overview of YANG Push. The purpose is to give you an understanding of how YANG Push works, but there are still plenty of details that are described in the RFCs that won't be covered in this application note. So, please, read the RFCs if you wish to know more details.

IETF's YANG Push work is comprised of a set of four RFCs:

- RFC 8639 – Subscription to YANG Notifications
- RFC 8640 – Dynamic Subscription to YANG Events and Datastores over NETCONF
- RFC 8650 – Dynamic Subscription to YANG Events and Datastores over RESTCONF
- RFC 8641 – Subscription to YANG Notifications for Datastore Updates

Because ConfD does not currently support YANG Push for RESTCONF or other transports, the focus of this overview will be YANG Push with NETCONF. ConfD's support for YANG Push will be discussed in a subsequent section.

The IETF NETCONF Working Group looked at NETCONF Notifications as defined in RFC 5277, NETCONF Event Notifications, as the basis for setting up subscriptions and delivering notifications to be used for YANG Push and realized there are several issues and limitations with RFC 5277. The basic idea of having a way to create a subscription to notifications and a way to send notifications was carried over from RFC 5277 to RFC 8639 and RFC 8640. One issue with RFC 5277 is that it is tied to NETCONF as a transport and there was an interest in being able to utilize other transports in addition to NETCONF for YANG Push. Another issue with RFC 5277 is that support needed to be added for Network Management Datastore Architecture (NMDA) as defined in RFC 8342. The final issue is that RFC 5277 pre-dated the YANG 1.0 standard. Thus, RFC 5277 defines an XML Schema and not a YANG data model.

3. Configured and Dynamic Subscriptions

RFC 8639 introduces two types of subscriptions, configured and dynamic. Configured subscriptions are modeled in YANG and created via a configuration operation, such as a NETCONF <edit-config> or <edit-data> operation or CLI. Dynamic subscriptions are similar to the subscription mechanism as described in RFC 5277 and use RPCs to create the subscription. Adding functionality which is not defined in RFC 5277, RFC 8639 also defines RPCs to modify a subscription and to explicitly delete a subscription. RFC 5277 relied on closing the underlying session as a way of implicitly ending the subscription.

RFC 8639 defines the RPCs <establish-subscription>, <modify-subscription>, <delete-subscription>, and <kill-subscription>.

The <establish-subscription> RPC from RFC 8639 allows for a number of choices as a target for the subscription. Later, we will see how RFC 8641 utilizes this choice, but, for now, RFC 8639 defines a stream as one choice, much as RFC 5277 specifies in its <create-subscription> RPC. The <establish-subscription> RPC also can take a filter as well as a stop-time, just as <create-subscription> allowed in RFC 5277. The <establish-subscription> RPC returns an id, which can be used in the three additional RPCs that are defined by RFC 8639. RFC 8639 also gives you the ability to use the <establish-subscription> RPC multiple times on a single transport session.

The <modify-subscription> RPC from RFC 8639 adds the capability to modify a subscription. In order to modify a subscription, you can, for example, specify a new filter to be used. This RPC must be issued on the same session that the <establish-subscription> RPC was sent on.

While you can still end a subscription implicitly by closing the transport connection, since you may have multiple subscriptions within a session, you may only want to close one of them. RFC 8639 defines the <delete-subscription> RPC, which takes the subscription id as input. This RPC is issued on the same transport session as the <establish-subscription> RPC was. You can also end a subscription on another transport session by using the <kill-subcription> operation.

4. Datastore Update

Now that we have covered a high level overview of the subscription support introduced in RFC 8639, it is natural to ask how is this used to support YANG Push? One clue is found in the title of RFC 8641, “Subscription to YANG Notifications for Datastore Updates”.

As you will recall from our earlier discussion, RFC 5277's <create-subscription> RPC built its notion of subscribing around a “stream”. In its definition of <establish-subscription>, RFC 8639 preserved this notion of subscribing to a “stream”, but also made "stream" simply one option in a YANG choice named "target".

RFC 8641 utilizes this choice to allow for another option to "stream" called "datastore". Hence, the title of RFC 8641. The use of datastore allows for the specification of which NMDA datastore we are interested in subscribing to.

5. Periodic and On-Change Subscriptions

In addition to the datastore we have selected, we also can choose whether our subscription to datastore objects is periodic or on-change.

For periodic subscriptions, you must specify a period for when the updates are sent. You can optionally specify an anchor time, which is used a reference point in time for calculating when the updates are to be sent. For example, if you want the notifications to be sent every 15 minutes at the normal 15-minute points on the clock, rather than when the <establish-subscription> was received, you would specify an anchor time that would give you the updates at those times.

For on-change subscriptions, a notification is sent whenever a change is detected in one of the subscribed objects. We know that changes can occur in rapid succession, such as the operational state of an interface when the interface flaps for some reason. Since these rapid changes may swamp the resources on the sender or receiver, you can specify a dampening

period. Successive updates on changes are only sent after the dampening period has passed. You can also specify the type of operation for which you want a notification. For example, you might only be interested in create or delete operations, but not when the value simply changes.

For both types of subscriptions, you can specify a filter to select the objects to be returned by the subscription. The two types of filter that can be specified are subtree filters and XPath filters. These filters are specified and applied as they are for other NETCONF operations.

In addition to the augments done on the RFC 8639 operations, the RFC 8641 YANG data model also defines a pair of notifications that are used to deliver on-change and periodic updates. A "push-update" notification is sent for periodic subscriptions. For on-change subscriptions, the "push-change-update" notification is used when changes occur in the subscribed objects. A "push-update" is also sent for on-change subscriptions when the user requests the initial value of the subscribed objects or wishes to synchronize the value of the subscription at some later point.

6. YANG Push Implementation in ConfD

Because RFCs 8639 and 8640 form the underlying support for RFC 8641, the ConfD product development team focused first on providing support for these two RFCs. Support for RFC 8639 and 8640 was added in ConfD 7.3, which was released in November 2019. ConfD continues to support RFC 5277 and the <create-subscription> RPC, so a NETCONF client can use either the older RFC 5277 style for subscribing or the new RPCs introduced by RFC 8639.

The mechanism for defining a stream in ConfD that can be used with RFC 8639 is the same as with RFC 5277. The stream is defined in the ConfD configuration file, confd.conf and here is an example:

```
<notifications>
  <eventStreams>
    <stream>
      <name>interface</name>
      <description>Example notifications</description>
      <replaySupport>true</replaySupport>
      <builtinReplayStore>
      <!-- enableBuiltinReplayStore -->
      <dir>./</dir>
      <maxSize>S1M</maxSize>
      <maxFiles>5</maxFiles>
      </builtinReplayStore>
    </stream>
  </eventStreams>
</notifications>
```

Here the stream is interface and this is the name that would specified in either the RFC 5277 <create-subscription> RPC or the RFC 8639 <establish-subscription> RPC.

How a ConfD application generates and sends a <notification> is the same no matter which method was used to start the subscription. First, a ConfD application invokes `confd_register_notification_stream()` specifying the stream name and, optionally, callback functions, if the application wishes to implement replay support. ConfD returns a notification context which can then be used to send <notifications> on that stream using the `confd_notification_send()` API call.

An example showing how RFC 5277 subscriptions work can be found in the ConfD examples under `$CONFD_DIR/examples.confd/netconf_notifications`. An example showing how RFC 8639 subscriptions work can be found under `$CONFD_DIR/examples.confd/netconf_notifications_rfc_8639`.

In ConfD 7.4, which was released in August 2020, support was added for periodic subscriptions and partial support was added for on-change subscriptions. Currently, only dynamic subscriptions are supported. Configured subscriptions are not supported. In ConfD 7.4, only on-change subscriptions are supported for the <running> datastore, not for the <operational> datastore.

For periodic subscriptions on the <operational> datastore, since this data is typically supplied by a ConfD data provider application, ConfD will invoke the data provider application when necessary to retrieve the current values of objects that the periodic subscription is interested in.

ConfD 7.5, which was released in January 2021, added support for on-change subscriptions on the <operational> datastore via an experimental API. Since there was no prior support in the ConfD Data Provider API (DPAPI) for a data provider application to notify ConfD when a value changed, a new API needed to be added. Since this API is new, the ConfD product development team decided to make the API experimental, meaning that the API may change in future releases, based on feedback and experience using the API. ConfD customers are encouraged to look at this API and provide feedback and suggestions which could improve the API.

One new API function call that has been added, `confd_register_push_on_change()`, is used for registering a pair of callback functions to handle new subscriptions and end existing subscriptions. Like other callback functions in ConfD, the signatures of these callback functions is specified by ConfD. Details of the signatures can be found in the ConfD confd_lib_dp(3) man page, in the section “PUSH ON-CHANGE CALLBACKS”.

Once the data provider application has received an on-change subscription request, and a change occurs, then a second new API function, `confd_push_on_change()`, is used to send the notification to the client who has subscribed to receive the on-change notification. Again, more information on how the applications supplies the information for `confd_push_on_change()` can be found in the ConfD confd_lib_dp(3) man page, in the section “PUSH ON-CHANGE CALLBACKS”.

An example showing how this experimental API works for on-change subscriptions as well as periodic subscriptions can be found in the ConfD examples under `$CONFD_DIR/examples.confd/netconf_yang_push`.

7. Conclusion

YANG Push is a set of RFCs developed by the IETF NETCONF Working Group to remove the traditional polling approach to periodically collect useful information from a network element and to introduce a more efficient means for a collector to receive the information.

With RFC 8639, a new transport agnostic approach to subscribing to notifications was introduced, as well as a set of extensible RPCs that can establish, modify, or end a subscription. RFC 8640 then defines how this can be carried out by NETCONF specifically. These RFCs can exist alongside the original NETCONF notification mechanism defined in RFC 5277.

RFC 8641 utilized the extensibility introduced by RFC 8639 to specify a new target for a subscription, namely a datastore, along with a means to select the objects in the datastore and whether the subscription is periodic or when a change occurs.

ConfD, over a series of releases, has added support for these RFCs to provide an implementation of YANG Push, including experimental support for on-change notifications on the <operational> datastore, which was added in ConfD 7.5. Additional YANG Push features will be supported in future releases of ConfD

8. For More Information

For more information about ConfD, visit https://www.tail-f.com

RFC 8639 – Subscription to YANG Notifications – https://tools.ietf.org/html/rfc8639

RFC 8640 – Dynamic Subscription to YANG Events and Datastores over NETCONF – https://tools.ietf.org/html/rfc8640

RFC 8650 – Dynamic Subscription to YANG Events and Datastores over RESTCONF – https://tools.ietf.org/html/rfc8650

RFC 8641 – Subscription to YANG Notifications for Datastore Updates – https://tools.ietf.org/html/rfc8641

tail-f a Cisco company

www.tail-f.com
info@tail-f.com

Corporate Headquarters

Sveavagen 25
111 34 Stockholm
Sweden
+46 8 21 37 40