Extensions to Yang Push
draft-ranade-netmod-yang-push-extension-02

Abstract

This document defines extensions to the yang push subscription mechanism, which can provide more granularity in tracking configuration changes in datastores.

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 August 9, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
1. Introduction

The subscription mechanism defined in yang-push draft
[I-D.ietf-netconf-yang-push] supports a subscription mechanism where
datstore and its datanodes can be provided as targets. However
there are a few scenarios where a subscription to datanodes may not
be sufficient.

- An application on the client, may want to track the configuration
  changes in a particular module. If a module contains "config
  true" mixed with "config false" nodes, setting up a filter which
  targets only "config true" nodes at multiple levels will be
cumbersome.

- An application on the client, may want to track the origin of the
  recently applied configuration. Yang-push subscription currently
does not provide such control.

1.1. Terminology

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

The following terms are defined in [RFC8342] and are not redefined
here:

- operational state datastore

- intended configuration datastore
2. Yang Push Subscription Extensions

There are two new filter nodes added to the Yang push subscription mechanism. They are listed below.

- config-filter
- with-origin

2.1. Config Filter

This is a filter for nodes with the given value for their 'config' property. When this leaf is set to 'true', only 'config true' nodes are selected and, when set to 'false', only 'config false' nodes are selected. If this leaf is not present, this filter is not applied.

2.2. Origin Metadata Attribute

This document defines the "with-origin" parameter, which if present, requests that the server includes "origin" metadata annotations as defined in [RFC8342] in notifications which are part of this subscription. This parameter is only valid for the operational state datastore and any datastores with identities derived from the "operational" identity. Otherwise, if an invalid datastore is specified then a suitable error is returned.

Data in the operational state datastore can come from multiple sources. The server should return the most accurate value for the "origin" metadata annotation as possible, indicating the source of the operational value, as specified in Section 5.3.4 of RFC8342 [1].

When encoding the origin metadata annotation for a hierarchy of returned nodes, the annotation may be omitted for a child node when the value matches that of the parent node, as described in the "ietf-origin" YANG module [RFC8342].

The "with-origin" parameter is optional to support. It is identified with the feature "origin".

3. Examples of usage

In this example, the following fictional module is used:
module example-interface {
    yang-version 1.1;
    namespace urn:example:interface;
    prefix int;

    import ietf-inet-types {
        prefix inet;
    }

    container interface {
        leaf name {
            type string;
        }

        leaf speed {
            type string;
            config false;
        }
    }
}

A subscription on the "config true" nodes of the operational state datastore. This subscription also requests to get the origin of the datanodes as part of Yang-push notifications. This is considering that the NETCONF protocol was used for this subscription.

<rpc xmlns="urn:ietf:params:xml:ns:netconf:1.0">
    <establish-subscription \
        xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-yang-push" \ 
        <yp:datastore>operational</yp:datastore>
        <ypext:config-filter>true</ypext:config-filter>
        <ypext:with-origin/>
    </establish-subscription>
</rpc>

The operator has configured an interface ("eth1"), the system has recognised an interface ("eth0") which was plugged in recently. The below NETCONF notification is an example for a yang-push notification when the device applies the "eth0" configuration.
<notification>
    xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
    <eventTime>2019-01-25T08:00:11.22Z</eventTime>
    <push-change-update
        xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push"
        xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin">
        <id>10</id>
        <datastore-changes>
            <yang-patch>
                <patch-id>1</patch-id>
                <edit>
                    <edit-id>edit1</edit-id>
                    <operation>create</operation>
                    <target>/example-interface:interface</target>
                    <value>
                        <interface xmlns="urn:example:interface"
                            or:origin="or:system">
                            <name>eth0</name>
                        </interface>
                    </value>
                </edit>
            </yang-patch>
        </datastore-changes>
    </push-change-update>
</notification>

4. YANG Module

<CODE BEGINS> file "ietf-yang-push-ext@2019-02-01.yang"
module ietf-yang-push-ext {
    yang-version 1.1;
    prefix ypext;

    import ietf-subscribed-notifications {
        prefix sn;
    }

    import ietf-yang-push {
        prefix yp;
    }

    import ietf-datastores {
        prefix ds;
        reference "RFC 8342: Network Management Datastore Architecture.";
    }

    organization "IETF NETMOD (Network Modeling) Working Group";
</CODE BEGINS>
This module defines extensions to subscription mechanism to yang-push on operational state datastore.

Copyright (c) 2019 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).

The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'MAY', and 'OPTIONAL' in the module text are to be interpreted as described in RFC 2119 (https://tools.ietf.org/html/rfc2119).

This version of this YANG module is part of RFC XXXX (https://tools.ietf.org/html/rfcXXXX); see the RFC itself for full legal notices.

revision 2019-02-01 {
  description
    "Initial revision.";
  reference "RFCXXXX";
}

feature origin {
  description
    "Indicates that the server supports the ‘origin’ annotation.";
  reference
    "RFC 8342: Network Management Datastore Architecture";
}

grouping subscription-filter-extension{
  description
    "This grouping describes the extensions to subscription
leaf config-filter {
  type boolean;
  description
    "Filter for nodes with the given value for their 'config'
    property. When this leaf is set to 'true', only 'config
    true' nodes are selected and, when set to 'false', only
    'config false' nodes are selected. If this leaf is not
    present, this filter is not applied.";
}

leaf with-origin {
  when 'derived-from-or-self(../yp:datastore, "ds:operational")';
  if-feature origin;
  type empty;
  description
    "If this parameter is present, the server will return
    the 'origin' annotation for the nodes that has one.";
}

augment "/sn:establish-subscription/sn:input/sn:target" + "/yp:datastore" {
  description
    "This augmentation adds the filter extensions for the
    subscription to RPC input.";
  uses subscription-filter-extension;
}

augment "/sn:modify-subscription/sn:input/sn:target" + "/yp:datastore" {
  description
    "This augmentation adds the filter extensions for the
    subscription to RPC input.";
  uses subscription-filter-extension;
}

augment "/sn:subscriptions/sn:subscription/sn:target" + "/yp:datastore" {
  description
    "This augmentation adds the filter extensions to
    selection filtering criteria.";
  uses subscription-filter-extension;
}

<CODE ENDS>
5. IANA Considerations

This document registers one URI in the IETF XML Registry [RFC3688]. The following registration has been made:

Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.

This document registers one YANG module in the YANG Module Names Registry [RFC6020]. The following registration has been made:

name: ietf-yang-push-ext
prefix: ypext
RFC: xxxx

6. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols 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 [RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

The origin metadata annotation exposes the origin of values in the applied configuration. Origin information may provide hints that certain control-plane protocols are active on a device. Since origin information is tied to applied configuration values, it is only accessible to clients that have the permissions to read the applied configuration values. Security administrators should consider the sensitivity of origin information while defining access control rules.
7. References

7.1. Normative References


7.2. URIs

[1] https://tools.ietf.org/html/rfc8342#section-5.3.4

Author’s Address

Rohit R Ranade
Huawei
Divyashree Techno Park, Whitefield
Bangalore, Karnataka  560066
India

Email: rohitrranade@huawei.com