NETCONF
Internet-Draft
Intended status: Standards Track
Expires: January 6, 2020

B. Lengyel
Ericsson
A. Clemm
Futurewei
B. Claise
Cisco Systems, Inc.
July 5, 2019

Yang-Push Notification Capabilities
draft-ietf-netconf-notification-capabilities-02

Abstract

This document proposes a YANG module that allows a server to specify server capabilities related to "Subscription to YANG Datastores" (Yang-Push). It proposes to use YANG Instance Data to document this information and make it already available at implementation time, but also allow it to be reported at runtime.

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 January 6, 2020.

Table of Contents

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 terms Yang-Push, On-change subscription and Periodic subscription are used as defined in [I-D.ietf-netconf-yang-push]

On-change Notification Capability: The capability of the server to support On-change subscriptions.

The term Server is used as defined in [RFC8342]

Implementation-time information: Information about the server's behavior that is made available during the implementation of the server, available from a source other then a running server.

Runtime-information: Information about the server's behavior that is available from the running server via management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040].

2. Introduction

As defined in [I-D.ietf-netconf-yang-push] a server may allow clients to subscribe to updates from a datastore and subsequently push such update notifications to the client. Notifications may be sent periodically or on-change (more or less immediately after each change).

A server supporting YANG-Push has a number of capabilities that are determined during the implementation of the server. These include:

- Supported (reporting) periods for periodic subscriptions
- Maximum number of objects that can be sent in an update

If the optional on-change feature is supported, these include:

- Supported dampening periods for on-change subscriptions
- The set of data nodes for which on-change notification is supported

Servers MAY have limitations in how many update notifications and how many datastore node updates they can send out in a certain time-period.

In some cases, a publisher supporting on-change notifications will not be able to push updates for some object types on-change. Reasons for this might be that the value of the datastore node changes frequently (e.g. in-octets counter), that small object changes are frequent and meaningless (e.g., a temperature gauge changing 0.1 degrees), or that the implementation is not capable of on-change notification for a particular object. In those cases, it will be important for client applications to have a way to identify for which objects on-change notifications are supported and for which ones not.

Faced with the reality that support for on-change notification does not mean that such notifications will be sent for any specific data node, client/management applications can not rely on the on-change functionality unless the client has some means to identify for which objects on-change notifications are supported. YANG models are meant to be used as an interface contract. Without identification of the data nodes actually supporting on-change, this contract would be incomplete.

This document proposes a YANG module that allows a client to discover YANG-Push related capabilities both at implementation-time and run-time.

Implementation time information is needed by Network Management System (NMS) implementers. During NMS implementation for any functionality that depends on the notifications the information about on change notification capability is needed. If the information is not available early in some document, but only as instance data from the network node once it is deployed, the NMS implementation will be delayed, because it has to wait for the network node to be ready. In addition, the assumption that all NMS implementers will have a correctly configured network node available to retrieve data from, is an expensive proposition and may not always hold. (An NMS may need to be able to handle many dozens of network node types.) Often a fully functional NMS is a requirement for introducing a new network node type into a network, so delaying NMS readiness effectively also delays the time at which a new network node type can be introduced into the network.

Implementation time information is needed by system integrators. When introducing a network node type into their network, operators often need to integrate the node type into their own management system. The NMS may have management functions that depend on on-change notifications. The network operator needs to plan his management practices and NMS implementation before he even decides to buy the specific network node type. Moreover the decision to buy the node type sometimes depends on these management possibilities.

Run-time information is needed:

- for any "purely model driven" client, e.g. a NETCONF-browser. As long as it has a valid model to read the capability information, it does not care which data nodes send notification, it will just handle what is available.

- in case the capability might change during run-time e.g. due to licensing, HW constraints etc.

- to check that early, implementation time capability information about the capabilities is indeed what the server implements (is the supplied documentation correct?)

3. Notification Capability Model

It is a goal to provide Yang-Push notification capability information in a format that is:

- vendor independent

- formal

- identical for implementation-time and run-time

The YANG module ietf-notification-capabilities is defined to provide the information. It contains:

a set of capabilities related to the amount of notifications the server can send out

a default on-change notification capability separately for config false and config true data nodes

an on-change-notification-capability list containing a potentially different true/false notification capability for a few data nodes in the schema tree. Unless a node is in this list with a specific capability value, it inherits its on-change-notification-capability from its parent in the data tree, or from the relevant default values. It is assumed that only a small number of nodes will be included in this list: special cases where the default behavior is not followed. For a detailed description of the usage of this list see the description in the YANG module.

The information SHALL be provided in two ways both following the ietf-notification-capabilities module:

- It SHALL be provided by the implementer as YANG instance data file complying to [I-D.ietf-netmod-yang-instance-file-format]. The file SHALL be available already in implementation time retrievable in a way that does not depend on a live network node. E.g. download from product Website.

- It SHALL be available via NETCONF [RFC6241] or RESTCONF [RFC8040] from the live server during runtime.

3.1. Tree Diagram

The following tree diagram [RFC8340] provides an overview of the data model.

```
module: ietf-notification-capabilities
  +--ro datastore-subscription-capabilities
     +--ro (update-period)?
     |  +--:(minimum-update-period)
     |  |  +--ro minimum-update-period?     uint32
     |  +--:(supported-update-period)
     |     +--ro supported-update-period*   uint32
     +--ro max-objects-per-update?          uint32
     +--ro minimum-dampening-period?        uint32 {yp:on-change}?
     +--ro on-change-capable-nodes* [datastore] {yp:on-change}?
        +--ro datastore                                union
        +--ro notification-sent-for-config-default?    boolean
        +--ro notification-sent-for-state-default?     boolean
        +--ro on-change-notification-capability* [node-selector]
           +--ro node-selector            nacm:node-instance-identifier
           +--ro on-change-supported      boolean
```

3.2. YANG Module

```
  <CODE BEGINS> file "ietf-notification-capabilities@2019-07-02.yang"

module ietf-notification-capabilities {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-notification-capabilities";
  prefix inc;

  import ietf-netconf-acm  { prefix nacm; }
  import ietf-yang-push    { prefix yp; }
  import ietf-yang-library {
    prefix yanglib;
    description
      "Requires revision 2019-01-04 or a revision derived from it.";
  }

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

     Editor:   Balazs Lengyel
               <mailto:balazs.lengyel@ericsson.com>";
  description
    "This module specifies YANG-Push related server
     capabilities.
```

```
      The module contains
      - capabilities related to the amount of notifications the
      server can send out. Note that for a specific subscription
      the server MAY still allow only longer periods or smaller
      updates depending on e.g. actual load conditions.
      - default and schema node specific information specifying
      the set of data nodes for which the server is capable
      of sending on-change notifications.

      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 (RFC 2119)
      (RFC 8174) when, and only when, they appear in all
      capitals, as shown here.

      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
      (http://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 2019-07-02 {
      description
        "Initial version";
      reference
        "RFC XXX: Yang-Push Notification Capabilities";
    }

    container datastore-subscription-capabilities {
      config false;
      description
        "YANG-Push related server capabilities";

      choice update-period {
        description
          "Supported period values.";
        leaf minimum-update-period {
          type uint32;
          units "centiseconds";
          description
```

```
          "Minimum update period supported for a
          periodic subscription. May be absent if the server is
          not capable of providing a specific value.";
      }

      leaf-list supported-update-period {
        type uint32;
        units "centiseconds";
        description
          "Specific supported update period values
          for a periodic subscription.";
      }
    }

    leaf max-objects-per-update {
      type uint32 {
            range "1..max";
      }
      description
        "Maximum number of objects that can be sent
         in an update. May be absent if the server is
          not capable of providing a specific value.";
    }

    leaf minimum-dampening-period {
      if-feature yp:on-change ;
      type uint32;
      units "centiseconds";
      description
        "The minimum dampening period supported for on-change
         subscriptions. May be absent if the server is
          not capable of providing a specific value.";
    }

    list on-change-capable-nodes {
      if-feature yp:on-change ;
      key datastore ;
      description "Specifies per datastore the data nodes for which the
        server is capable of sending on-change notifications.
        If a datastore implemented by the server is not specified
        in this list and there is no list element for 'all' datastores
        the datastore does not support any on-change notifications.

        On-change notification capability is marked as true or false.
        This marking is inherited from the parent down the data tree
        unless explicitly marked otherwise.

        On-change notifications SHALL be sent for a config=true
```

```
        data node if one of the following is true:
        - if it is a top level data-node and is not specified in the
        on-change-notification-capability list and the
        notification-sent-for-config-default is true; or
        - notifications are sent for its parent data node and it is
        not specified in the on-change-notification-capability list; or
        - it is specified in the on-change-notification-capability
        list and has an on-change-supported value true.

        On-change notifications SHALL be sent for a config=false
        data node if one of the following is true:
        - if it is a top level data-node (a config=false data node with
        a config=true parent SHALL be treated as a top level data node)
        and is not specified in the on-change-notification-capability
        list and the notification-sent-for-state-default is true; or
        - notifications are sent for its parent data node
        which is also config=false and it is
        not specified in the on-change-notification-capability list; or
        - it is specified in the on-change-notification-capability
        list and has an on-change-supported value true";

      leaf datastore {
        type union {
          type leafref {
            path /yanglib:yang-library/yanglib:datastore/yanglib:name ;
          }
          type enumeration {
            enum all ;
          }
        }
        must '. != "all" or count(..) = "1" ' {
          error-message
            "If 'all' is present individual datastores cannot be " +
              "specified.";
        }
        description "The datastore for which on-change capable
          nodes are defined.";
      }

      leaf notification-sent-for-config-default {
        type boolean;
        default "true";
        description
          "Specifies the default value for
           top level configuration data nodes for the
           on-change-supported capability.";
      }
```

```
        leaf notification-sent-for-state-default {
          type boolean;
          default "false";
          description
            "Specifies the default value
             top level state data nodes for the
             on-change-supported capability.";
        }

        list on-change-notification-capability {
          key "node-selector";
          description
            "A list of data nodes that have the
             on-change-notification-capability specifically defined.

             Should be used only when specific data nodes support
             on-change notification in a module/subtree that
             generally does not support it or when some data nodes
             do not support the notification in a module/subtree
             that generally supports on-change notifications.";

          leaf node-selector {
            type nacm:node-instance-identifier;
            description
              "Selects the data nodes for which
               on-change capability is specified.";
          }

          leaf on-change-supported {
            type boolean;
            mandatory true;
            description
              "Specifies whether the server is capable of
               sending on-change notifications for the selected
               data nodes.";
          }
        }
      }
    }
  }
```

<CODE ENDS>

4. 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 data in this module is not security sensitive.

5. IANA Considerations

5.1. The IETF XML Registry

This document registers one URI in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested:

URI: urn:ietf:params:xml:ns:yang:ietf-notification-capabilities
Registrant Contact: The NETCONF WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

5.2. The YANG Module Names Registry

This document registers one YANG module in the YANG Module Names registry [RFC7950]. Following the format in [RFC7950], the the following registrations are requested:

```
name:       ietf-notification-capabilities
namespace:  urn:ietf:params:xml:ns:yang:ietf-notification-capabilities
prefix:     inc
reference:  RFC XXXX
```

6. Open Issues

-

7. References

7.1. Normative References

[I-D.ietf-netconf-yang-push]
Clemm, A. and E. Voit, "Subscription to YANG Datastores", draft-ietf-netconf-yang-push-25 (work in progress), May 2019.

[I-D.ietf-netmod-yang-instance-file-format]
Lengyel, B. and B. Claise, "YANG Instance Data File Format", draft-ietf-netmod-yang-instance-file-format-02 (work in progress), February 2019.

[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>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[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>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

7.2. Informative References

[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>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

Appendix A. Instance data examples

The following example is instance-data describing the notification capabilities of a hypothetical "acme-switch". The switch implements the running, candidate and operational datastores. Every change can be reported on-change from running, nothing from candidate and all config=false data from operational.

```
<?xml version="1.0" encoding="UTF-8"?>
<instance-data-set xmlns=
    "urn:ietf:params:xml:ns:yang:ietf-yang-instance-data">
  <name>acme-switch-notification-capabilites</name>
  <!-- Content-schema specification, revision date, contact, etc. -->
  <description>Notification capabilities of acme-switch.
    Acme-switch implements the running, candidate and operational
    datastores. Every change can be reported on-change from running,
    nothing from candidate and all config=false data from operational.
      </description>
  <content-data>
    <datastore-subscription-capabilities>
      <minimum-update-period>500</minimum-update-period>
      <max-objects-per-update>2000</max-objects-per-update>
      <minimum-dampening-period>100</minimum-dampening-period>
      <on-change-capable-nodes>
        <datastore xmlns="urn:ietf:params:xml:ns:yang:ietf-datastores">
          running
        </datastore>
        <!-- Neither notification-sent-for-config-default or
        notification-sent-for-state-default are present as the
        default values are in effect. -->
      </on-change-capable-nodes>
      <!-- The candidate datastore is implemented, but not present
      here as it does not support any on-change notifications. -->
      <on-change-capable-nodes>
        <datastore xmlns="urn:ietf:params:xml:ns:yang:ietf-datastores">
          operational
        </datastore>
        <notification-sent-for-config-default>
          false
        </notification-sent-for-config-default>
        <notification-sent-for-state-default>
          true
        </notification-sent-for-state-default>
      </on-change-capable-nodes>
    </datastore-subscription-capabilities>
  </content-data>
</instance-data-set>
```

Figure 1: Notification Capabilities with default settings

The following is the instance-data describing the notification capabilities of a hypothetical "acme-router". The router implements the running, and operational datastores. Every change can be reported on-change from running, but only config=true nodes and some config=false data from operational. Interface statistics are not reported on-change only 2 important counters.

```
<?xml version="1.0" encoding="UTF-8"?>
<instance-data-set xmlns=
    "urn:ietf:params:xml:ns:yang:ietf-yang-instance-data">
  <name>acme-router-notification-capabilites</name>
  <!-- Content-schema specification, revision date, contact, etc. -->
  <description>Defines the notification capabilities of an acme-router.
    The router only has running, and operational datastores.
    Every change can be reported on-change from running, but
    only config=true nodes and some config=false data from operational.
    Statistics are not reported on-change only 2 important counters.
      </description>
  <content-data>
    <datastore-subscription-capabilities
     xmlns="urn:ietf:params:xml:ns:yang:ietf-notification-capabilities">
      <minimum-update-period>500</minimum-update-period>
      <max-objects-per-update>2000</max-objects-per-update>
      <minimum-dampening-period>100</minimum-dampening-period>
      <on-change-capable-nodes>
        <datastore xmlns="urn:ietf:params:xml:ns:yang:ietf-datastores">
          running
        </datastore>
      </on-change-capable-nodes>
      <on-change-capable-nodes>
        <datastore xmlns="urn:ietf:params:xml:ns:yang:ietf-datastores">
          operational
        </datastore>
        <notification-sent-for-state-default>
          true
        </notification-sent-for-state-default>
        <on-change-notification-capability>
          <node-selector>
            /if:interfaces/if:interface/if:statistics</node-selector>
          <on-change-supported>false</on-change-supported>
        </on-change-notification-capability>
        <on-change-notification-capability>
          <node-selector>
            /if:interfaces/if:interface/if:statistics/if:in-octets
          </node-selector>
          <on-change-supported>true</on-change-supported>
        </on-change-notification-capability>
        <on-change-notification-capability>
          <node-selector>
            /if:interfaces/if:interface/if:statistics/if:out-octets
          </node-selector>
          <on-change-supported>true</on-change-supported>
        </on-change-notification-capability>
      </on-change-capable-nodes>
    </datastore-subscription-capabilities>
```

```
  </content-data>
</instance-data-set>
```

Figure 2: Notification Capabilities with data node specific settings

Appendix B. Changes between revisions

v01 - v02

- Added instance data examples
- On-change capability can be defined per datastore
- Added "if-feature yp:on-change" where relevant
- Unified units used

v00 - v01

- Add more capabilities: minimum period, supported period max-number of objects, min dampening period, dampening supported

Authors' Addresses

Balazs Lengyel
Ericsson
Magyar Tudosok korutja 11
1117 Budapest
Hungary

Email: balazs.lengyel@ericsson.com

Alexander Clemm
Futurewei
2330 Central Expressway
Santa Clara, CA 95050
USA

Email: ludwig@clemm.org

Benoit Claise
Cisco Systems, Inc.
De Kleetlaan 6a b1
1831 Diegem
Belgium

Email: bclaise@cisco.com