Abstract

This document proposes a YANG module that allows a publisher to specify 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 run-time.

The YANG module is also prepared to contain other system capabilities, for future augmentations.

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 June 11, 2020.

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
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 [RFC8641]

The terms Subscriber, Publisher and Receiver are used as defined in [RFC8639]

The term Server is used as defined in [RFC8342]

On-change Notification Capability: The capability of the publisher to send on-change notifications for a specific datastore or a specific data node.

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

Run-time information: Information about the publisher’s behavior that is available from the running server (implementing the publisher) via management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040].

2. Introduction

As defined in [RFC8641] a publisher may allow subscribers to subscribe to updates from a datastore and subsequently push such update notifications to the receiver. Notifications may be sent periodically or on-change (more or less immediately after each change).

A publisher supporting YANG-Push has a number of capabilities defined in [RFC8641] that are often determined during the implementation of the publisher. These include:

- Supported (reporting) periods for periodic subscriptions
- Maximum number of objects that can be sent in an update
- The set of datastores or data nodes for which periodic notification is supported

If the optional on-change feature is supported, additionally:

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

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

Publishers might not support periodic subscriptions to all datastores.

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 irrelevant to the receiver (e.g., a temperature gauge changing 0.1 degrees within a predetermined and acceptable range), or that the implementation is not capable of on-change notification for a particular object. In those cases, it will be important for...
subscriber applications to have a way to identify 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, subscriber/management applications can not rely on the on-
change functionality unless the subscriber has some means to identify
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 subscriber to
discover YANG-Push related capabilities both at implementation-time
and run-time.

Implementation-time information is needed by Network Management
System (NMS) implementers. A NMS implementation that wants to
support notifications, needs the information about on-change
notification capability. If the information is not documented in a
way available to the NMS designer, 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" application, e.g. a NETCONF-browser.
  Such applications depend on reading models, capabilities in run-
time to support all the publisher’s available functionality.
in case the capability might change during run-time e.g. due to licensing, HW constraints etc.

to check that capability information provided early, already in implementation-time is indeed what the publisher implements (is the supplied documentation correct?)

The proposed YANG module is also intended as a base model to be augmented by other YANG modules defining system-capabilities not related to YANG-Push. Other capability defining YANG modules MAY augment the data nodes specifying these capabilities into this model.

3. Notification Capability Model

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

- vendor independent
- machine readable
- 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 throughput of notification data the publisher can send out.
- specification of which data nodes support on-change notifications.

Capability values can be specified on system/publisher level, datastore level or on specific data nodes (and their contained sub-tree) of a specific datastore. Capability values on a smaller, more specific part of the publisher’s data always override more generic values.

Note: The solution is usable for both NMDA and non-NMDA systems. For non-NMDA servers/publishers the config=false data is considered as if it was part of the running datastore.

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

- For the implementation-time use-case: It SHOULD 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.

- For the run-time use-case: It SHOULD be available via NETCONF [RFC6241] or RESTCONF [RFC8040] from the live server (implementing the publisher) during run-time. Implementations which support changing these capabilities at run-time SHOULD support on-change notifications about the system-capabilities container.

### 3.1. Tree Diagram

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

```
module: ietf-notification-capabilities
  +--ro system-capabilities
    +--ro 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-supported? notification-support (yp:on-change)?:
      +--ro periodic-notifications-supported? notification-support
      +--ro supported-excluded-change-type* union {yp:on-change}?
    +--ro datastore-capabilities* [datastore]
      +--ro datastore -> /yanglib:yang-library/datastore/name
    +--ro per-node-capabilities* [node-selector]
      +--ro node-selector nacm:node-instance-identifier
      +--ro 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
        |   +--ro on-change-supported? notification-support {yp:on-change}?:
        |   +--ro periodic-notifications-supported? notification-support
        +--ro supported-excluded-change-type* union {yp:on-change}?
```

3.2. YANG Module

<CODE BEGINS> file "ietf-notification-capabilities@2019-12-09.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;
    description
      "This module does not require NACM to be implemented, as
      only a NACM typedef is used";
  }

  import ietf-yang-push {
    prefix yp;
    description
      "This module requires ietf-yang-push to be implemented for the
      two subscription-capabilities containers.";
  }

  import ietf-yang-library {
    prefix yanglib;
    description "This module requires ietf-yang-library to
    be implemented. Revision 2019-01-04 or a
    revision derived from it is required.";
  }

  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 publisher
    capabilities.

    The module contains
    - capabilities related to the throughput of notification data the
      publisher can support. (Note that for a specific subscription
      the publisher MAY still allow only longer periods or smaller
      updates depending on e.g. actual load conditions.)
    - specification of which data nodes support on-change or periodic
Capability values can be specified on system/publisher level, datastore level or on specific data nodes (and their contained sub-tree) of a specific datastore. If a capability is specified on multiple levels, the specification on a more specific level overrides more generic capability specifications; thus
- a system/publisher level specification is overridden by any other specification
- a datastore level specification (with a node-selector '/') is overridden by a specification with a more specific node-selector.
- a specification for a specific datastore and node-selector is overridden by a specification for the same datastore with a node-selector that describes more levels of containing lists and containers.

If, different data nodes covered by a single subscription have different values for a specific capability, then using values that are only acceptable for some of these data nodes, but not for others, may result in the rejection of the subscription.

To find a capability value for a specific node in a specific datastore the user SHALL
1) consider the system/publisher level capabilities under the system-capabilities container if the capability value is specified.
2) search for a datastore-capabilities list entry for the specific datastore.
3) within that datastore entry search for a per-node-capabilities entry that specifies the specific capability and that has the node-selector selecting the specific data node and that specifies the most levels of containing containers and lists.
4) If no entries are found in the previous steps the publisher is not capable of providing a value because it is unknown, the capability is changing for some reason, there is no specified limit etc. In this case the publisher’s behavior is unspecified.

revision 2019-12-09 {
  description
    "Initial version";
  reference
    "RFC XXX: YANG-Push Notification Capabilities";
}

grouping subscription-capabilities {
  description "Capabilities related to Yang-Push notifications";
  container subscription-capabilities {
    description "Capabilities related to Yang-Push notifications";
    typedef notification-support {
      type enumeration {
        enum no-notifications-supported {
          description "The publisher is not capable of sending any
                        notifications for the relevant scope and subscription
                        type.";
        }
        enum notifications-for-config-changes-supported {
          description "The publisher is capable of sending
                        notifications for config=true nodes, but not
                        for config=false nodes for the relevant scope
                        and subscription type.";
        }
        enum notifications-for-state-changes-supported {
          description "The publisher is capable of sending
                        notifications for config=false nodes, but not
                        for config=true nodes for the relevant scope
                        and subscription type.";
        }
        enum notifications-for-all-changes-supported {
          description "The publisher is capable of sending
                        notifications for both config=false and config=true
                        nodes for the relevant scope and subscription type.";
        }
      }
    }
  }
}
description "Type for defining whether on-change or
periodic notifications are supported for no, only config=true,
only config=false or all data nodes."
)

choice update-period {
  description "Supported update period value or values for
  periodic subscriptions."
  leaf minimum-update-period {
    type uint32;
    units "centiseconds";
    description "Indicates the minimal update period that is
    supported for a periodic subscription.
    A periodic subscription to the selected data nodes must
    specify a value that is at least as large or greater than
    this";
    reference
      "The period leaf in RFC 8641 ietf-yang-push YANG module";
  }
  leaf-list supported-update-period {
    type uint32;
    units "centiseconds";
    description "Supported update period values for a
    periodic subscription.
    A periodic subscription to the selected data nodes must
    specify one of the values in the list; other values
    are not supported.";
    reference
      "The period leaf in RFC 8641 ietf-yang-push YANG module";
  }
}

leaf max-objects-per-update {
  type uint32 {
    range "1..max";
  }
  description
    "Maximum number of objects that can be sent
    in an update for the selected data nodes.";
}

leaf minimum-dampening-period {
  if-feature yp:on-change;
  type uint32;
  units "centiseconds";
description
"The minimum dampening period supported for on-change
subscriptions for the selected data nodes."
}

leaf on-change-supported {
  if-feature yp:on-change;
  type notification-support;
  description
"Specifies whether the publisher is capable of
sending on-change notifications for the selected
data store or data nodes and the subtree below them.";
}

leaf periodic-notifications-supported {
  type notification-support;
  description
"Specifies whether the publisher is capable of
sending periodic notifications for the selected
data store or data nodes and the subtree below them.";
}

leaf-list supported-excluded-change-type {
  if-feature yp:on-change;
  type union {
    type enumeration {
      enum none {
        description "None of the change types can be excluded.";
      }
      enum all {
        description
"Any combination of change types can be excluded.";
      }
    }
    type yp:change-type;
  }
  description "The change types that can be excluded in
YANG-Push subscriptions.";
}
}

container system-capabilities {
  config false;
  description "YANG-Push related publisher capabilities.
  Capability values specified here at the publisher level
  are valid for all datastores and
  are used when the capability is not specified on the
datastore level or for specific data nodes.

uses subscription-capabilities {
  refine subscription-capabilities/supported-excluded-change-type {
    default none;
  }
}
}

list datastore-capabilities {
  key datastore;

description "Capabilities values per datastore. For non-NMDA servers/publishers the config=false data is considered as if it was part of the running datastore."

leaf datastore {
  type leafref {
    path /yanglib:yang-library/yanglib:datastore/yanglib:name;
  }

description "The datastore for which capabilities are defined. Only individual datastores can be specified e.g. ds:conventional is not allowed.";
}

list per-node-capabilities {
  key "node-selector";

description "Each list entry specifies notification capabilities for the selected data nodes. The same capabilities apply for the data nodes in the subtree below them unless another list entry with a more specific node selector is present."

leaf node-selector {
  type nacm:node-instance-identifier;

description "Selects the data nodes for which capabilities are specified. The special value '/’ denotes all data nodes in the datastore. The system SHOULD order list entries according to the tree structure of the data models to make reading/parsing the data model more simple."
}

uses subscription-capabilities;
}
3.3. Other System Capabilities

Other YANG modules defined in separate documents MAY augment this module to define other capabilities not related to YANG-Push. Every set of such capabilities SHOULD be wrapped in a container similar to the subscription-capabilities container in this YANG module to cleanly separate different groups of capabilities. The "other-capabilities" container SHOULD be augmented as a sibling to the subscription-capabilities container.

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.

All protocol-accessible data nodes are read-only and cannot be modified. The data in this module is not security sensitive. Access control may be configured, to avoid exposing the read-only data.

When that data is in file format, data should be protected against modification or unauthorized access using normal file handling mechanisms.

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:

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 following registrations are requested:

name:       ietf-notification-capabilities
prefix:     inc
reference:  RFC XXXX

6. References

6.1. Normative References

[I-D.ietf-netmod-yang-instance-file-format]


6.2. Informative References


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. Periodic subscriptions are supported for running and operational, but not for candidate.

```xml
<?xml version="1.0" encoding="UTF-8"?>
  <name>acme-switch-notification-capabilities</name>
  <yid-version>1</yid-version>
  <content-schema>
    <module>ietf-notification-capabilities@2019-12-04</module>
  </content-schema>
</instance-data-set>
```

Lengyel, et al. Expires June 11, 2020
datastores. Every change can be reported on-change from running, nothing from candidate and all config=false data from operational. Periodic subscriptions are supported for running and operational, but not for candidate.
</description>
<content-data>
<system-capabilities
 xmlns="urn:ietf:params:xml:ns:yang:ietf-notification-capabilities"
<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>
 <periodic-notifications-supported>
  notifications-for-all-changes-supported
 </periodic-notifications-supported>
</subscription-capabilities>
<datastore-capabilities>
 <datastore>ds:operational</datastore>
 <per-node-capabilities>
  <node-selector></node-selector>
  <subscription-capabilities>
   <on-change-supported>
    notifications-for-state-changes-supported
   </on-change-supported>
  </subscription-capabilities>
 </per-node-capabilities>
</datastore-capabilities>
<datastore-capabilities>
 <datastore>ds:candidate</datastore>
 <per-node-capabilities>
  <node-selector></node-selector>
  <subscription-capabilities>
   <on-change-supported>no-notifications-supported
   </on-change-supported>
   <periodic-notifications-supported>
    no-notifications-supported
   </periodic-notifications-supported>
  </subscription-capabilities>
 </per-node-capabilities>
</datastore-capabilities>
<datastore-capabilities>
 <datastore>ds:running</datastore>
 <per-node-capabilities>
  <node-selector></node-selector>
  <subscription-capabilities>
   <on-change-supported>notifications-for-all-changes-supported
   </on-change-supported>
  </subscription-capabilities>
 </per-node-capabilities>
</datastore-capabilities>
</system-capabilities>
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. Datastore subscription capabilities are not reported on-change as they never change on the acme-router during run-time.

```xml
  <name>acme-router-notification-capabilities</name>
  <yid-version>1</yid-version>
  <content-schema>
    <module>ietf-notification-capabilities@2019-12-04</module>
  </content-schema>
  <!-- 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, for these a smaller dampening period is possible.</description>
</instance-data-set>

<system-capabilities
  xmlns="urn:ietf:params:xml:ns:yang:ietf-notification-capabilities"
  <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>
    <per-node-capabilities>
    </per-node-capabilities>
    <datastore-capabilities>
    </datastore-capabilities>
  </subscription-capabilities>
</system-capabilities>
```

Figure 1: Notification Capabilities with datastore level settings
Figure 2: Notification Capabilities with data node specific settings
Appendix B. Changes between revisions

v07 - v08

- Prepared the YANG model to include other non-YANG-Push related capabilities.
- Renamed the top level container to system-capabilities.
- Added a container subscription-capabilities to the grouping subscription-capabilities to contain all subscription related capabilities.
- Updated examples according to draft-ietf-netmod-yang-instance-file-format-06.

v06 - v07

- Updated examples according to draft-ietf-netmod-yang-instance-file-format-05.

v05 - v06

- Providing the capability data is only a "SHOULD" recommendation. Some reviewers wanted MUST some wanted much less.
- The YANG module import statements now indicate the imported modules that must be implemented not just available as import as requested by the YangDoctors review.

v04 - v05

- Added new capabilities periodic-notifications-supported and supported-excluded-change-type.
- Restructured YANG module to make the node-selector’s usage similar to how NACM uses it: "/" means the whole datastore.
- Small corrections, spelling, rewording
- Replaced the term server with the term publisher except in cases where we speak about datastores and functionality based on get, getconfig operations. In this latter case it is really the server functionality that is discussed.

v03 - v04
Clarified recommended support for on-change notifications about the datastore-subscription-capabilities.

v02 - v03

- Allow throughput related capabilities to be defined on top, datastore or data node level. Described that specific capability values always override generic ones.
- Indicate that non-NMDA servers can also use this model.
- Updated according to draft-ietf-netmod-yang-instance-file-format-04

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