Generic YANG-related System Capabilities and YANG-Push Notification Capabilities
draft-ietf-netconf-notification-capabilities-09

Abstract

This document proposes two YANG modules. The module ietf-system-capabilities provides a structure that can be used to specify any YANG related system capability.

The module ietf-notification-capabilities 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.

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 July 10, 2020.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.
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 or server’s behavior that is made available during the implementation of the publisher/server, available from a source other than a running server.

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

2. Introduction

Systems implementing a server and/or a publisher often have capabilities that are not defined by the YANG model itself. There is a need to publish this capability information as part of the contract between the server and client. Examples include: maximum size of data that can be stored or transferred, information about counters (whether a node supports on-change telemetry), etc. Such capabilities are often dependent on a vendor’s implementation or the available resources at deployment. Many such capabilities are specific to either the complete system, individual YANG datastores or specific parts of the YANG schema, or even to individual data nodes. It is a goal of this document to provide a common way of representing such capabilities in a format that is:

- vendor independent
- machine readable
- available in identical format both in implementation-time and run-time

2.1. YANG-Push Notification Capabilities

A specific case where we need to specify capabilities is the YANG-Push functionality. 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

Additional capabilities if the optional on-change feature is supported include:

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.

Clients of a server, subscribers to a publisher need a method to gather capability information.

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 and 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?)

3. Providing System Capability Information

Capability information is represented by instance-data based on one or more "capability defining YANG modules". This allows a user to discover capabilities both at implementation-time and run-time.

- For the implementation-time use-case: Capabilities SHOULD be provided by the implementer as YANG instance data files 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: Capabilities 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.

The module ietf-system-capabilities is defined to provide a structure that can be used to specify any YANG related system capability.

The module ietf-notification-capabilities is defined to allow a publisher to specify capabilities related to "Subscription to YANG Datastores" (YANG-Push) augmenting ietf-system-capabilities.

4. System Capabilities Model

The module ietf-system-capabilities is defined to provide a structure that can be used to specify any YANG related system capability.

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

This module itself does not contain any capabilities. It SHOULD be used by other modules to augment-in specific capability information. Every set of such capabilities SHOULD be wrapped in a container under the augment statement to cleanly separate different groups of capabilities. These "wrapper containers" SHALL be augmented in at /sysc:system-capabilities and /sysc:system-capabilities/sysc:datastore-capabilities/sysc:per-node-capabilities.

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

4.1. Tree Diagram

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

module: ietf-system-capabilities
  +--ro system-capabilities
    +--ro datastore-capabilities* [datastore]
      +--ro datastore -> /yanglib:yang-library/datastore/name
    +--ro per-node-capabilities* [node-selector]
      +--ro node-selector nacm:node-instance-identifier
4.2. YANG Module

<CODE BEGINS> file "ietf-system-capabilities@2020-01-02.yang"

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

  import ietf-netconf-acm {
    prefix nacm;
    description "The module ietf-netconf-acm is OPTIONAL to implement."
  }

  import ietf-yang-library {
    prefix yanglib;
    description "The module ietf-yang-library is REQUIRED 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 a module intended to contain system capabilities. System capabilities may include capabilities of a NETCONF or RESTCONF server or a notification publisher.

  This module does not contain any specific capabilities it only provides a structure where containers containing the actual capabilities should be augmented in.

  Capability values can be specified on system level, datastore level or for 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 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.

It is not allowed to have multiple node selectors which
- are defined for the same datastore AND
- have the same number of containment levels AND
- select an overlapping set of nodes.

To find a capability value for a specific data node in a specific datastore the user SHALL
1) consider the system 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 system’s behavior is unspecified.


Copyright (c) 2020 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 (https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself.
for full legal notices.

revision 2020-01-02 {
  description
    "Initial version";
  reference
    "RFC XXX: YANG-Push Notification Capabilities";
}

container system-capabilities {
  config false;
  description "System capabilities.
    Capability values specified here at the system level
    are valid for all datastores and
    are used when the capability is not specified on the
    datastore level or for specific data nodes."
} // augmentation point for system level capabilities

list datastore-capabilities {
  key datastore;

  description "Capabilities values per datastore.
    For non-NMDA servers/publishers 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 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 specifying the same
         capability 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 more simple.

// augmentation point for datastore or data node level
// capabilities


5. Notification Capabilities Model

The YANG module ietf-notification-capabilities is defined to provide YANG-Push related capability information.

5.1. Tree Diagram

The following tree diagram [RFC8340] provides an overview of the data model.
module ietf-notification-capabilities
augment /sysc: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}?

augment /sysc:system-capabilities/sysc:datastore-capabilities/
  +
  | sysc:per-node-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}?

5.2. YANG Module

<CODE BEGINS> file "ietf-notification-capabilities@2020-01-02.yang"

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

  import ietf-yang-push {
    prefix yp;
    description
      "The module ietf-yang-push is REQUIRED to be implemented.";
  }

  import ietf-system-capabilities {


Internet-Draft     YANG-Push Notification Capabilities      January 2020

prefix sysc;
description
    "The module ietf-system-capabilities is REQUIRED to be
     implemented.";
}

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
- specification of which data nodes support on-change or periodic
  notifications.
- 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.)

Capability values can be specified on system/publisher level,
datastore level or for specific data nodes (and their contained
sub-tree) of a specific datastore, as defined in the
ietf-system-capabilities module.

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.

The key words ‘MUST’, ‘MUST NOT’, ‘REQUIRED’, ‘SHALL’,
‘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) 2020 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


This version of this YANG module is part of RFC XXXX (https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself for full legal notices.

revision 2020-01-02 {
  description "Initial version";
  reference "RFC XXX: YANG-Push Notification Capabilities";
}

grouping subscription-capabilities {
  description "Capabilities related to YANG-Push subscriptions and notifications";
  container subscription-capabilities {
    description "Capabilities related to YANG-Push subscriptions and 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.";
      }
    } yp:change-type;
  }
  description "The change types that can be excluded in YANG-Push subscriptions.";
}

augment /sysc:system-capabilities {
  description "Add system level capabilities";
  uses subscription-capabilities {
    refine subscription-capabilities/supported-excluded-change-type {
      default none;
    }
  }
}
augment "/sysc:system-capabilities/sysc:datastore-capabilities" + 
"/sysc:per-node-capabilities" {
    description "Add datastore and node level capabilities";
    uses subscription-capabilities {
        refine subscription-capabilities/supported-excluded-change-type {
            default none;
        }
    }
}

<CODE ENDS>

6. Security Considerations

The YANG modules specified in this document define 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 these modules 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.

7. IANA Considerations

7.1. The IETF XML Registry

This document registers two URIs 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.
7.2. The YANG Module Names Registry

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

name:       ietf-system-capabilities
prefix:     sysc
reference:  RFC XXXX

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

8. References

8.1. Normative References

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


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 version="1.0" encoding="UTF-8"?>
<instance-data-set xmlns=""..."
<name>acme-switch-notification-capabilities</name>
<yid-version>1</yid-version>
<content-schema>
  <module>ietf-system-capabilities@2020-01-02</module>
  <module>ietf-notification-capabilities@2020-01-02</module>
</content-schema>
<!-- 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.
  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-system-capabilities"
    xmlns:inc="urn:ietf:params:xml:ns:yang:ietf-notification-capabilities"
    <inc:minimum-update-period>500</inc:minimum-update-period>
    <inc:max-objects-per-update>2000</inc:max-objects-per-update>
    <inc:minimum-dampening-period>100</inc:minimum-dampening-period>
    <inc:periodic-notifications-supported>
      notifications-for-all-changes-supported
    </inc:periodic-notifications-supported>
  </inc:subscription-capabilities>
  <datastore-capabilities>
    <datastore>ds:operational</datastore>
    <per-node-capabilities>
      <node-selector>/</node-selector>
      <inc:subscription-capabilities>
        <inc:on-change-supported>
          notifications-for-state-changes-supported
        </inc:on-change-supported>
      </inc:subscription-capabilities>
    </per-node-capabilities>
  </datastore-capabilities>
  <datastore-capabilities>
    <datastore>ds:candidate</datastore>
    <per-node-capabilities>
      <node-selector>/</node-selector>
      <inc:subscription-capabilities>
        <inc:on-change-supported>no-notifications-supported
        </inc:on-change-supported>
        <inc:periodic-notifications-supported>
      </inc:subscription-capabilities>
    </per-node-capabilities>
  </datastore-capabilities>
</system-capabilities>
</content-data>
Figure 1: Notification Capabilities with datastore level 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. Datastore subscription capabilities are not reported on-change as they never change on the acme-router during run-time.

<?xml version="1.0" encoding="UTF-8"?>
   name="acme-router-notification-capabilities"
   yid-version="1"
   content-schema="ietf-system-capabilities@2020-01-02"
   ietf-notification-capabilities@2020-01-02"
   <!-- 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>
<content-data>
  <system-capabilities
      xmlns="urn:ietf:params:xml:ns:yang:ietf-system-capabilities"
      xmlns:inc="urn:ietf:params:xml:ns:yang:ietf-notification-capabilities"
    <inc:subscription-capabilities>
      <inc:minimum-update-period>500</inc:minimum-update-period>
      <inc:max-objects-per-update>2000</inc:max-objects-per-update>
      <inc:minimum-dampening-period>100</inc:minimum-dampening-period>
      <inc:periodic-notifications-supported>
        notifications-for-all-changes-supported
      </inc:periodic-notifications-supported>
      <inc:on-change-supported>
        notifications-for-all-changes-supported
      </inc:on-change-supported>
      <inc:supported-excluded-change-type>
        all
      </inc:supported-excluded-change-type>
    </inc:subscription-capabilities>
    <datastore-capabilities>
      <datastore>ds:operational</datastore>
      <per-node-capabilities>
        <node-selector>
          /if:interfaces/if:interface/if:statistics
        </node-selector>
        <inc:subscription-capabilities>
          <inc:on-change-supported>
            no-notifications-supported
          </inc:on-change-supported>
        </inc:subscription-capabilities>
      </per-node-capabilities>
      <per-node-capabilities>
        <node-selector>
          /if:interfaces/if:interface/if:statistics/if:in-octets
        </node-selector>
        <inc:subscription-capabilities>
          <inc:minimum-dampening-period>10
        </inc:minimum-dampening-period>
          <inc:on-change-supported>
            notifications-for-all-changes-supported
          </inc:on-change-supported>
        </inc:subscription-capabilities>
      </per-node-capabilities>
      <per-node-capabilities>
        <node-selector>
          /if:interfaces/if:interface/if:statistics/if:out-octets
        </node-selector>
        <inc:subscription-capabilities>
          <inc:on-change-supported>
            notifications-for-all-changes-supported
          </inc:on-change-supported>
        </inc:subscription-capabilities>
      </per-node-capabilities>
    </per-node-capabilities>
  </system-capabilities>
</content-data>
<inc:minimum-dampening-period>10</inc:minimum-dampening-period>
<inc:on-change-supported>
  notifications-for-all-changes-supported
</inc:on-change-supported>
<inc:subscription-capabilities>
</per-node-capabilities>
</datastore-capabilities>
</system-capabilities>
</content-data>
</instance-data-set>

Figure 2: Notification Capabilities with data node specific settings

Appendix B. Changes between revisions

v08 - v09

- Split the YANG module into two: ietf-system-capabilities and ietf-notification-capabilities. Restructured/updated the draft accordingly.

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
o 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 bl
1831 Diegem
Belgium

Email: bclaise@cisco.com