NETCONF
Internet-Draft
Intended status: Standards Track
Expires: September 10, 2020

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

March 9, 2020

Generic YANG-related System Capabilities and YANG-Push Notification Capabilities
draft-ietf-netconf-notification-capabilities-12

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

Copyright Notice

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 [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 then 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 it 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:

o vendor independent

o machine readable

o 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:

o 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.

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

o 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.

o 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 (by selecting all nodes in the datastore) or for specific data nodes of a specific datastore (and their contained sub-trees). Capability values specified for a specific datastore or node-set override values specified on the system/publisher level.

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* []
            +--ro (node-selection)?
               +--:(node-selector)
                  +--ro node-selector?   nacm:node-instance-identifier
```

4.2. YANG Module

This YANG module imports typedefs from [RFC8341] and a reference path from [RFC8525].

```
<CODE BEGINS> file "ietf-system-capabilities@2020-03-08.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;
  }
  import ietf-yang-library {
    prefix yanglib;
    description
      "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 (by selecting all nodes in the datastore) or
     for specific data nodes of a specific datastore (and their
     contained sub-trees).
     Capability values specified for a specific datastore or
     node-set override values specified on the system/publisher level.

     To find a capability value for a specific data node in a
     specific datastore the user SHALL
     1) search for a datastore-capabilities list entry for
     the specific datastore.
     2) If the datastore entry is found within that entry process all
     per-node-capabilities entries in the order they appear in the list.
     The first entry that specifies the specific capability and has a
     node-selector selecting the specific data node defines the
     capability value.
     3) If the capability value is not found above and the specific
     capability is specified under the system-capabilities container
     (outside the datastore-capabilities list) this value shall be used.
     4) If no values are found in the previous steps the
     system/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.

     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) 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
     (https://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 2020-03-08 {
    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 {
        description
          "Each list entry specifies capabilities for the selected
           data nodes. The same capabilities apply for the data nodes
           in the subtree below the selected nodes.
           The system SHALL order the entries according to their
           precedence. The order of the entries MUST NOT change unless
           the underlying capabilities also change.";
        choice node-selection {
          description
            "A method to select all or some nodes within a datastore.";
          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.";
          }
```

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

   <CODE ENDS>
```

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-nodes-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-nodes-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

This YANG module imports a feature and typedefs from [RFC8641].

```
<CODE BEGINS> file "ietf-notification-capabilities@2020-03-09.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;
  }
  import ietf-system-capabilities {
    prefix sysc;
  }

  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 [RFC 8641] 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 of a specific
     datastore (and their contained sub-trees), 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',
        '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) 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 2020-03-09 {
       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 bits {
             bit config-changes {
               description
                 "The publisher is capable of sending
                  notifications for config=true nodes for the relevant
                  scope and subscription type.";
             }
             bit state-changes {
               description
```

```
                "The publisher is capable of sending
                 notifications for config=false nodes for the relevant
                 scope and subscription type.";
            }
          }
          description
            "Type for defining whether on-change or
             periodic notifications are supported for none, only
             config=true, only config=false or all data nodes.

             If the bit config-changes or state-changes is set
             for a datastore or a set of nodes that does not contain
             nodes with the indicated config value,
             this has no effect, as if no support was declared.
             E.g. indicating support for state-changes for
             a candidate datastore has no effect.";
        }

        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 subscription request to the selected data
               nodes with a smaller period than what this leaf
               specifies will result in a 'period-unsupported' error.";
            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 subscription request to the selected data nodes with a
               period not included in the leaf-list will result in a
               'period-unsupported' error.";
            reference
              "The period leaf in RFC 8641 ietf-yang-push YANG module";
          }
        }
        leaf max-nodes-per-update {
```

```
           type uint32 {
             range "1..max";
           }
           description
             "Maximum number of data nodes that can be sent
              in an update. The publisher MAY support more data nodes,
              but SHOULD support at least this number.
              May be used to avoid the update-too-big error
              during subscription.";
           reference "The update-too-big error/identity in RFC 8641";
         }
         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 {
                 value -2;
                 description
                   "None of the change types can be excluded.";
               }
               enum all {
                 value -1;
                 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.";
         }
       }
     }

     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:

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

   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.
```

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 the following registrations are requested:

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

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

8. References

8.1. Normative References

[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-07 (work in progress), February 2020.

[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>.

[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>.

[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>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[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>.

[RFC8525] Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", RFC 8525, DOI 10.17487/RFC8525, March 2019, <https://www.rfc-editor.org/info/rfc8525>.

[RFC8639] Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Subscription to YANG Notifications", RFC 8639, DOI 10.17487/RFC8639, September 2019, <https://www.rfc-editor.org/info/rfc8639>.

[RFC8641] Clemm, A. and E. Voit, "Subscription to YANG Notifications for Datastore Updates", RFC 8641, DOI 10.17487/RFC8641, September 2019, <https://www.rfc-editor.org/info/rfc8641>.

8.2. Informative References

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[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>.

[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>.

[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>.

[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>.

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=
    "urn:ietf:params:xml:ns:yang:ietf-yang-instance-data">
  <name>acme-switch-notification-capabilities</name>
  <yid-version>1</yid-version>
  <content-schema>
    <module>ietf-system-capabilities@2020-03-08</module>
    <module>ietf-notification-capabilities@2020-03-09</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"
        xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
        <inc:subscription-capabilities>
          <inc:minimum-update-period>500</inc:minimum-update-period>
          <inc:max-nodes-per-update>2000</inc:max-nodes-per-update>
          <inc:minimum-dampening-period>100</inc:minimum-dampening-period>
          <inc:periodic-notifications-supported>
            config-changes state-changes
          </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>
                state-changes
              </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/>
              <inc:periodic-notifications-supported/>
            </inc:subscription-capabilities>
          </per-node-capabilities>
        </datastore-capabilities>
        <datastore-capabilities>
          <datastore>ds:running</datastore>
          <per-node-capabilities>
            <node-selector>/</node-selector>
            <inc:subscription-capabilities>
              <inc:on-change-supported>
                config-changes
              </inc:on-change-supported>
```

```
          </inc:subscription-capabilities>
        </per-node-capabilities>
      </datastore-capabilities>
    </system-capabilities>
  </content-data>
</instance-data-set>
```

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"?>
<instance-data-set xmlns=
    "urn:ietf:params:xml:ns:yang:ietf-yang-instance-data">
  <name>acme-router-notification-capabilities</name>
  <yid-version>1</yid-version>
  <content-schema>
    <module>ietf-system-capabilities@2020-03-08</module>
    <module>ietf-notification-capabilities@2020-03-09</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>
  <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"
      xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
      <inc:subscription-capabilities>
        <inc:minimum-update-period>500</inc:minimum-update-period>
        <inc:max-nodes-per-update>2000</inc:max-nodes-per-update>
        <inc:minimum-dampening-period>100</inc:minimum-dampening-period>
        <inc:periodic-notifications-supported>
          config-changes state-changes
        </inc:periodic-notifications-supported>
```

```
        <inc:on-change-supported>
          config-changes state-changes
        </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:name='lo']
          </node-selector>
          <inc:subscription-capabilities>
            <inc:on-change-supported/>
            <inc:periodic-notifications-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>
              state-changes
            </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:minimum-dampening-period>10
              </inc:minimum-dampening-period>
            <inc:on-change-supported>
              state-changes
            </inc:on-change-supported>
          </inc:subscription-capabilities>
        </per-node-capabilities>
        <per-node-capabilities>
          <node-selector>
              /if:interfaces/if:interface/if:statistics
          </node-selector>
          <inc:subscription-capabilities>
            <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

v11 - v12

- Updated max-nodes-per-update description
- Reformatted YANG models with pyang -f yang --keep-comments --yang-line-length 69

v10 - v11

- Updated examples
- Updated typedef notification-support

v09 - v10

- Removed description text from imports about the need for implementing the imported module.
- Changed notification-support to bits with shorter names
- Assigned enum values to supported-excluded-change-type
- Made node-selector a choice to allow for future alternative selection methods.
- Changed precedence of per-node-capabilities entries. Precedence is now according to the position of entries in the list.

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

- o Allow throughput related capabilities to be defined on top, datastore or data node level. Described that specific capability values always override generic ones.

- o Indicate that non-NMDA servers can also use this model.

- o Updated according to draft-ietf-netmod-yang-instance-file-format-04

v01 - v02

- o Added instance data examples

- o On-change capability can be defined per datastore

- o Added "if-feature yp:on-change" where relevant

- o 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 b1
1831 Diegem
Belgium

Email: bclaise@cisco.com