Workgroup: OPSAWG
Internet-Draft:
draft-ietf-opsawg-collected-data-manifest-02
Published: 23 October 2023
Intended Status: Standards Track
Expires: 25 April 2024
Authors: B. Claise J. Quilbeuf D. Lopez
Huawei Huawei Telefonica I+D
I. Dominguez T. Graf
Telefonica I+D Swisscom

A Data Manifest for Contextualized Telemetry Data

Abstract

Network elements use Model-driven Telemetry, and in particular YANG-Push, to continuously stream information, including both counters and state information. This document documents the metadata that ensure that the collected data can be interpreted correctly. This document specifies the Data Manifest, composed of two YANG data models (the Platform Manifest and the Data Collection Manifest). The Data Manifest must be streamed and stored along with the data, up to the collection and analytics system in order to keep the collected data fully exploitable by the data scientists.

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 25 April 2024.

Table of Contents

1. Introduction

Network elements use Model-driven Telemetry (MDT), and in particular YANG-Push [RFC8641], to continuously stream information, including both counters and state information.

This document specifies what needs to be kept as metadata (i.e., the Data Manifest) to ensure that the collected data can still be interpreted correctly throughout the collection and network analytics toolchain. When streaming YANG-structured data with YANG-Push

[RFC8641], there is a semantic definition in the corresponding YANG module definition. This is the semantic information for the collected objects: While this semantic is absolutely required to correctly decode and interpret the data, understanding the network element and collection environment contexts information is equally important to interpret the data.

This document proposes the Data Manifest, which is composed of two YANG data models, namely, the Platform Manifest and the Data Collection Manifest, in order to keep the collected data exploitable by the data scientists.

The Platform Manifest contains information characterizing the platform streaming the telemetry information, while the the Data Collection Manifest contains the required information to characterize how and when the telemetry information was metered.

The two proposed YANG modules in the Data Manifest do not expose many new information but rather define what should be exposed by a platform streaming telemetry. Some related YANG modules have been specified to retrieve the platform capabilities:

* The IETF YANG Library [RFC8525].

* YANG Modules Describing Capabilities for Systems and Datastore Update Notifications [RFC9196] for the platform capabilities regarding the production and export of telemetry data.

* [I-D.claise-netconf-metadata-for-collection], which is based on the previous draft to define the optimal settings to stream specific items (i.e., per path).

These related YANG modules are important to discover the capabilities before applying the telemetry configuration (such as on-change). Some of their content is part of the context for the streamed data.

We first present the module for the Platform Manifest in Section 3 and then the module for the Data Collection Manifest in Section 4. The full Data Manifest is obtained by combining these two modules. We explain in Section 5 how the Data Manifest can be retrieved and how collected data is mapped to the Data Manifest.

1.1. Use Cases

1.1.1. Network Analytics

Streamed information from network elements is used for network analytics, incident detections, and in the end closed-loop automation. This streamed data can be stored in a database (sometimes called a big data lake) for further analysis.

As an example, a database could store a time series representing the evolution of a specific counter collected from a network element. When analyzing the data, the network operator/data scientist must understand the context information for these data:

*This object definition in the YANG model.

*The network element specific vendor, platform, and OS.

*The collection parameters.

Characterizing the source used for producing the data (vendor, platform, and OS) is useful to complement the data. As an example, knowing the exact data source software specification might reveal a particularity in the observed data, explained by a specific bug, a specific bug fix, or simply a particular specific behavior. This is also necessary to ensure the reliability of the collected data. On top of that, in particular for YANG-Push [RFC8641], it is crucial to know the set of YANG modules supported by the platform, along with their deviations. In some cases, there might even be some backwards incompatible changes in native modules between one OS version to the next one. This information is captured by the proposed Platform Manifest.

From a collection parameters point of view, the data scientists analyzing the collected data must know that the counter was requested from the network element as on-change or at specific cadence. Indeed, an on-change collection explains why there is a single value as opposed to a time series. In case of periodic collection, this exact cadence might not be observable in the time series. Indeed, this time series might report some values as 0 or might even omit some values. The reason for this behavior might be diverse: the network element was under stress, with a too small observation period, compared to the minimum-observed-period [I-D.claise-netconf-metadata-for-collection]. Again, knowing the conditions under which the counter was collected and streamed (along with the platform details) help drawing the right conclusions. As an example, taking into account the value of 0 might lead to a wrong conclusion that the counter dropped to zero. This document specifies the Data Collection Manifest, which contains the required information to characterize how and when the telemetry information was metered.

The goal of the current document is to define what needs to be kept as metadata (i.e., the Data Manifest) to ensure that the collected data can still be interpreted correctly.

1.1.2. New Device Onboarding

When a new device is onboarded, operators must make sure that the new device streams data with YANG-Push, that the telemetry data is the

right ones, that the data is correctly ingested in the collection system, and finally that the data can be analyzed (compared with other similar devices). For the last point, the Data Manifest, which must be linked to the data up to the collection and analytics system, contains all the relevant information.

1.1.3. Data Mesh Principles in Networking

The concept behind the data mesh https://www.datamesh-architecture.com/ are:

* Principle of Domain Ownership: Architecturally and organizationally align business, technology, and analytical data, following the line of responsibility. Here, the Data Mesh principles adopt the boundary of bounded context to individual data products where each domain is responsible for (and owns) its data and models.

* Principle of Data as a Product: The “Domain” owners are responsible to provide the data in useful way (discoverable through a catalog, addressable with a permanent and unique address, understandable with well defined semantics, trustworthy and truthful, self-describing for easy consumption, interoperable by supporting standards, secure, self-contained, etc.) and should treat consumers of that data as customers. It requires and relies on the “Domain Ownership” principle.

* Principle of Self-serve Data Platform: This fosters the sharing of cross-domain data in order to create extra value.

* Principle of Federated Computational Governance: Describes the operating model and approach to establishing global policies across a mesh of data products.

The most relevant concept for this document is the "Data as a Product" principle. The Data Manifest fulfills this principle as the two YANG data models, Platform Manifest and the Data Collection Manifest, along with the data, provide all the necessary information in a self-describing way for easy consumption.

2. 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.

Data Manifest: all the necessary data required to interpret the telemetry information.

Platform Manifest: part of the Data Manifest that completely characterizes the platform producing the telemetry information

Data Collection Manifest: part of the Data Manifest that completely characterizes how and when the telemetry information was metered.

3. Platform Manifest

3.1. Overview of the Model

Figure 1 contains the YANG tree diagram [RFC8340] of the ietf-platform-manifest module.

```
module: ietf-platform-manifest
  +--ro platforms
     +--ro platform* [id]
        +--ro id                    string
        +--ro name?                 string
        +--ro vendor?               string
        +--ro vendor-pen?           uint32
        +--ro software-version?     string
        +--ro software-flavor?      string
        +--ro os-version?           string
        +--ro os-type?              string
        +--ro yang-push-streams
        |  +--ro stream* [name]
        |     +--ro name            string
        |     +--ro description?    string
        +--ro yang-library
           +--ro module-set* [name]
           |  +--ro name                   string
           |  +--ro module* [name]
           |  |  +--ro name              yang:yang-identifier
           |  |  +--ro revision?         revision-identifier
           |  |  +--ro namespace         inet:uri
           |  |  +--ro location*         inet:uri
           |  |  +--ro submodule* [name]
           |  |  |  +--ro name              yang:yang-identifier
           |  |  |  +--ro revision?         revision-identifier
           |  |  |  +--ro location*         inet:uri
           |  |  |  +--ro revision-label?   rev:revision-label
           |  |  +--ro feature*          yang:yang-identifier
           |  |  +--ro deviation*        -> ../../module/name
           |  |  +--ro revision-label?   rev:revision-label
           |  +--ro import-only-module* [name revision]
           |     +--ro name              yang:yang-identifier
           |     +--ro revision          union
           |     +--ro namespace         inet:uri
           |     +--ro location*         inet:uri
           |     +--ro submodule* [name]
           |     |  +--ro name              yang:yang-identifier
           |     |  +--ro revision?         revision-identifier
           |     |  +--ro location*         inet:uri
           |     |  +--ro revision-label?   rev:revision-label
           |     +--ro revision-label?   rev:revision-label
           +--ro schema* [name]
           |  +--ro name                            string
           |  +--ro module-set*
           |  |       -> ../../module-set/name
           |  +--ro deprecated-nodes-implemented?   boolean
           |  +--ro obsolete-nodes-absent?          boolean
           +--ro datastore* [name]
```

```
          +--ro name      ds:datastore-ref
          +--ro schema    -> ../../schema/name
```

Figure 1: YANG tree diagram for ietf-platform-manifest module

The YANG module actually contains a list of Platform Manifests (in 'platforms/platform'), indexed by the identifier of the platform. That identifier should be defined by the network manager so that each platform has a unique id. As an example, the identifier could be the 'sysname' from the ietf-notification module presented in [I-D.tgraf-netconf-notif-sequencing]. The scope of this module is the scope of the data collection, i.e. a given network, therefore it contains a collection of Platform Manifests, as opposed to the device scope, which would contain a single Platform Manifest.

The Platform Manifest is identified by a set of parameters ('name', 'software-version', 'software-flavor', 'os-version', 'os-type') that are aligned with the YANG Catalog www.yangcatalog.org [I-D.clacla-netmod-model-catalog] so that the YANG Catalog could be used to retrieve the YANG modules a posteriori. The vendor of the platform can be identified via its name 'vendor' or its PEN number 'vendor-pen', as described in [RFC9371].

In order to provide information for yang-push subscriptions based on streams, the Platform Manifest specifies the streams available on the platform within the 'yang-push-streams' container. That container is similar to the one from the ietf-subscribed-notifications module, and the Data Collection Manifest uses it to refer to streams used by subscriptions.

The Platform Manifest also includes the contents of the YANG Library [RFC8525]. That module set is particularly useful to define the paths, as they are based on module names. Similarly, this module defines the available datastores, which can be referred to from the Data Manifest, if necessary. If supported by the platform, fetching metrics from a specific datastore could enable some specific use cases: monitoring configuration before it is committed, comparing between the configuration and operational datastore.

3.2. YANG module ietf-platform-manifest

<CODE BEGINS> file "ietf-platform-manifest@2023-03-08.yang"

```
module ietf-platform-manifest {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-platform-manifest";
  prefix p-mf;

  import ietf-yang-library {
    prefix yanglib;
    reference
      "RFC8525: YANG Library";
  }
  import ietf-yang-revisions {
    prefix rev;
    reference
      "XXXX: Updated YANG Module Revision Handling";
  }

  organization
    "IETF OPSAWG (Network Configuration) Working Group";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/opsawg/>
     WG List:  <mailto:opsawg@ietf.org>
     Author:   Benoit Claise   <mailto:benoit.claise@huawei.com>
     Author:   Jean Quilbeuf   <mailto:jean.quilbeuf@huawei.com>
     Author:   Diego R. Lopez  <diego.r.lopez@telefonica.com>
     Author:   Ignacio Dominguez
                          <ignacio.dominguezmartinez@telefonica.com>
     Author:   Thomas Grapf     <thomas.graf@swisscom.com>";
  description
    "This module describes the platform information to be used as
     context of data collection from a given network element. The
     contents of this model must be streamed along with the data
     streamed from the network element so that the platform context
     of the data collection can be retrieved later.

     The data content of this model should not change except on
     upgrade or patching of the device.

     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) 2022 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 Revised 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; see the
   RFC itself for full legal notices.  ";

revision 2023-03-08 {
  description
    "Initial revision";
  reference
    "RFC xxxx: Title to be completed";
}

// This is a copy-paste of the augments from the module
// ietf-yang-library-revisions.
// We just changed the path to which the data is added

augment "/p-mf:platforms/p-mf:platform/p-mf:yang-library"
      + "/p-mf:module-set/p-mf:module" {
  description
    "Add a revision label to module information";
  leaf revision-label {
    type rev:revision-label;
    description
      "The revision label associated with this module revision.
       The label MUST match the rev:revision-label value in the
       specific revision of the module loaded in this module-set.";
    reference
      "XXXX: Updated YANG Module Revision Handling;
       Section 5.2.1, Advertising revision-label";
  }
}

augment "/p-mf:platforms/p-mf:platform/p-mf:yang-library/"
      + "p-mf:module-set/p-mf:module/p-mf:submodule" {
  description
    "Add a revision label to submodule information";
  leaf revision-label {
    type rev:revision-label;
    description
      "The revision label associated with this submodule revision.
       The label MUST match the rev:revision-label value in the
       specific revision of the submodule included by the module
       loaded in this module-set.";
    reference
      "XXXX: Updated YANG Module Revision Handling;
       Section 5.2.1, Advertising revision-label";
  }
```

```
}

augment
  "/p-mf:platforms/p-mf:platform/p-mf:yang-library/p-mf:module-set/"
+ "p-mf:import-only-module" {
  description
    "Add a revision label to module information";
  leaf revision-label {
    type rev:revision-label;
    description
      "The revision label associated with this module revision.
       The label MUST match the rev:revision-label value in the
       specific revision of the module included in this module-set.";
    reference
      "XXXX: Updated YANG Module Revision Handling;
       Section 5.2.1, Advertising revision-label";
  }
}

augment
  "/p-mf:platforms/p-mf:platform/p-mf:yang-library/p-mf:module-set/"
+ "p-mf:import-only-module/p-mf:submodule" {
  description
    "Add a revision label to submodule information";
  leaf revision-label {
    type rev:revision-label;
    description
      "The revision label associated with this submodule revision.
       The label MUST match the rev:label value in the specific
       revision of the submodule included by the
       import-only-module loaded in this module-set.";
    reference
      "XXXX: Updated YANG Module Revision Handling;
       Section 5.2.1, Advertising revision-label";
  }
}

augment "/p-mf:platforms/p-mf:platform/p-mf:yang-library"
      + "/p-mf:schema" {
  description
    "Augmentations to the ietf-yang-library module to indicate how
     deprecated and obsoleted nodes are handled for each datastore
     schema supported by the server.";
  leaf deprecated-nodes-implemented {
    type boolean;
    description
      "If set to true, this leaf indicates that all schema nodes with
       a status 'deprecated' are implemented
       equivalently as if they had status 'current'; otherwise
```

```
        deviations MUST be used to explicitly remove deprecated
        nodes from the schema.  If this leaf is absent or set to false,
        then the behavior is unspecified.";
      reference
        "XXXX: Updated YANG Module Revision Handling;
        Section 5.2.2, Reporting how deprecated and obsolete nodes
        are handled";
    }
    leaf obsolete-nodes-absent {
      type boolean;
      description
        "If set to true, this leaf indicates that the server does not
        implement any status 'obsolete' schema nodes.  If this leaf is
        absent or set to false, then the behaviour is unspecified.";
      reference
        "XXXX: Updated YANG Module Revision Handling;
        Section 5.2.2, Reporting how deprecated and obsolete nodes
        are handled";
    }
  }

  container platforms {
    config false;
    description
      "Top container including all platforms in scope. If this model is
      hosted on a single device, it should contain a single entry in
      the list. At the network level, it should contain an entry for
      every monitored platform.";
    list platform {
      key "id";
      description
        "Contains information about the platform that allows to identify
        and understand the individual data collection information.  ";
      leaf id {
        type string;
        description
          "Identifies a given platform on the network, for instance the
          sysname of the plaftorm. The id has to be unique on the
          network.";
      }
      leaf name {
        type string;
        description
          "Model of the platform from which data is collected.";
      }
      leaf vendor {
        type string;
        description
          "Organization that implements that platform.";
```

```
}
leaf vendor-pen {
  type uint32;
  description
    "Vendor''s registered Private Enterprise Number as
     described in RFC9371";
}
leaf software-version {
  type string;
  description
    "Name of the version of software. With respect to most network
     device appliances, this will be the operating system version.
     But for other YANG module implementation, this would be a
     version of appliance software. Ultimately, this should
     correspond to a version string that will be recognizable by
     the consumers of the platform.";
}
leaf software-flavor {
  type string;
  description
    "A variation of a specific version where YANG model support
     may be different.  Depending on the vendor, this could be a
     license, additional software component, or a feature set.";
}
leaf os-version {
  type string;
  description
    "Version of the operating system using this module. This is
     primarily useful if the software implementing the module is
     an application that requires a specific operating system
     version.";
}
leaf os-type {
  type string;
  description
    "Type of the operating system using this module. This is
     primarily useful if the software implementing the module is
     an application that requires a specific operating system
     type.";
}
container yang-push-streams {
  // copied from "streams" container of
  // ietf-subscribed-notifications
  config false;
  description
    "Contains information on the built-in event streams provided
     by the platform.";
  list stream {
    key "name";
```

```
          description
            "Identifies the built-in event streams that are supported by
             the publisher.";
          leaf name {
            type string;
            description
              "A handle for a system-provided event stream made up of a
               sequential set of event records, each of which is
               characterized by its own domain and semantics.";
          }
          leaf description {
            type string;
            description
              "A description of the event stream, including such
               information as the type of event records that are
               available in this event stream.";
          }
        }
      }
      container yang-library {
        description
          "The YANG library of the device specifying the modules
           available in each of the datastores.";
        uses yanglib:yang-library-parameters;
      }
    }
  }
}
```

<CODE ENDS>

4. Data Collection Manifest

4.1. Overview of the Model

Figure 2 contains the YANG tree diagram [RFC8340] of the ietf-data-collection-manifest module.

```
module: ietf-data-collection-manifest
  +--ro data-collections
     +--ro data-collection* [platform-id]
        +--ro platform-id
        |       -> /p-mf:platforms/platform/id
        +--ro yang-push-subscriptions
           +--ro subscription* [id]
              +--ro id
              |       sn:subscription-id
              +--ro (target)
              |  +--:(stream)
              |  |  +--ro stream                            leafref
              |  |  +--ro (filter-spec)?
              |  |     +--:(stream-subtree-filter)
              |  |     |  +--ro stream-subtree-filter?      <anydata>
              |  |     |          {subtree}?
              |  |     +--:(stream-xpath-filter)
              |  |        +--ro stream-xpath-filter?
              |  |                yang:xpath1.0 {xpath}?
              |  +--:(datastore)
              |     +--ro datastore                         leafref
              |     +--ro (datastore-filter-spec)?
              |        +--:(datastore-subtree-filter)
              |        |  +--ro datastore-subtree-filter?   <anydata>
              |        |          {sn:subtree}?
              |        +--:(datastore-xpath-filter)
              |           +--ro datastore-xpath-filter?
              |                   yang:xpath1.0 {sn:xpath}?
              +--ro transport?
              |       sn:transport
              +--ro encoding?
              |       sn:encoding
              +--ro purpose?                                string
              +--ro dscp?                                   inet:dscp
              |       {dscp}?
              +--ro weighting?                              uint8
              |       {qos}?
              +--ro dependency?
              |       subscription-id {qos}?
              +--ro (update-trigger)?
              |  +--:(periodic)
              |  |  +--ro periodic!
              |  |     +--ro period         centiseconds
              |  |     +--ro anchor-time?   yang:date-and-time
              |  +--:(on-change) {on-change}?
              |     +--ro on-change!
              |        +--ro dampening-period?   centiseconds
              +--ro current-period?
              |       yp:centiseconds
```

```
            +--ro receivers
               +--ro receiver* [name]
                  +--ro name                      string
                  +--ro sent-event-records?
                  |       yang:zero-based-counter64
                  +--ro excluded-event-records?
                  |       yang:zero-based-counter64
                  +--ro state                     enumeration
```

Figure 2: YANG tree diagram for ietf-data-collection-manifest module

The 'data-collections' container contains the information related to each YANG-Push subscription. As for the Platform Manifest, these subscriptions are indexed by the platform id, so that all subscriptions in the network can be represented in the module.

The YANG-Push collection is organized in subscriptions, the parameters for such a subscription are specified in [RFC8639] and [RFC8641]. The list of subscriptions from a given platform is stored in '/data-collection/data-collection/yang-push-subscriptions/ subscription'. Subscription metadata are the bulk of the Data Collection Manifest, they are heavily based on the two RFCs above.

The 'target' choice specify the selected contents for the subscription. We did not include the target with a reference to a common filter as stored in 'filters' in ietf-subscribed-notfications. The rationale for this choice is that otherwise we would need to store these filters in the Platform Manifest, which could cause changes in that manifest more often than needed. If a stream based subscription is used [RFC8639], the stream must exist in the 'yang-push-streams' container of Platform Manifest, which is modelled as a leafref in our module. If a datastore based subscription is used [RFC8641], the datastore must exist in the 'yang-library' container of the Platform Manifest, which is modelled by a leafref as well.

We also included 'transport', 'encoding' and 'purpose' as in ietf-subscribed-notification, but without the feature switches as we are not concerned about using this module for configuration.

We also included the 'dscp', 'weighting' and 'dependency' as in ietf-subscribed-notification, again without the feature switches. This information might be useful to understand why the collection is failing or less frequent than expected for a given subscription.

The 'update-trigger' choice from ietf-yang-push is included as well, as it is crucial to understand the frequency at which notifications should arrive.

The only new content is the 'current-period', which might differ from the requested period (when in periodic collection mode) if the

platform implements a mechanism to increase the collection period when it is overloaded.

Finally, we also included the state of the receivers for that subscription, as in ietf-subscribed-notifications.

This information is crucial to understand the collected values. For instance, the 'on-change' trigger, if used, might remove a lot of messages from the database because values are sent only when there is a change.

4.2. YANG module ietf-data-collection-manifest

<CODE BEGINS> file "ietf-data-collection-manifest@2023-03-08.yang"

```
module ietf-data-collection-manifest {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-data-collection-manifest";
  prefix d-mf;

  import ietf-platform-manifest {
    prefix p-mf;
    reference
      "RFC XXXX: Title to be completed";
  }
  import ietf-subscribed-notifications {
    prefix sn;
    reference
      "RFC 8639: A YANG Data Model for Subscriptions to
         Event Notifications";
  }
  import ietf-yang-push-modif {
    prefix yp;
    reference
      "RFC 8641: Subscriptions to YANG Datastores. TODO fix and
       used original version. This import is a modified version so that
       it compiles.";
  }
  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991: Common YANG Data Types";
  }

  organization
    "IETF OPSAWG (Network Configuration) Working Group";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/opsawg/>
     WG List:  <mailto:opsawg@ietf.org>
     Author:   Benoit Claise   <mailto:benoit.claise@huawei.com>
     Author:   Jean Quilbeuf   <mailto:jean.quilbeuf@huawei.com>
     Author:   Diego R. Lopez  <diego.r.lopez@telefonica.com>
     Author:   Ignacio Dominguez
                          <ignacio.dominguezmartinez@telefonica.com>
     Author:   Thomas Grapf     <thomas.graf@swisscom.com>";
  description
    "This module describes the context of data collection from a
     given network element. The contents of this model must be
     streamed along with the data streamed from the network
     element so that the context of the data collection can
     be retrieved later.

     This module must be completed with
```

```
   ietf-platform-manifest
   to capture the whole context of a data collection session.

   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) 2022 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 Revised 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; see the
   RFC itself for full legal notices.  ";

revision 2023-03-08 {
  description
    "Initial revision";
  reference
    "RFC xxxx: Title to be completed";
}

container data-collections {
  config false;
  description
    "Contains the configuration and statistics for the collected data,
     per node in the network.";
  list data-collection {
    key "platform-id";
    description
      "Defines the information for each collected object";
    leaf platform-id {
      type leafref {
        path "/p-mf:platforms/p-mf:platform/p-mf:id";
      }
      description
        "Id of the platform collecting the data. This id is the same
         as the one in the platform manifest.";
    }
    container yang-push-subscriptions {
      /*
       * Copy-pasting here some of the contents of the "subscriptions"
```

```
 * container from ietf-subscribed notification.
 */
description
  "Contains the list of currently active subscriptions, i.e.,
   subscriptions that are currently in effect, used for
   subscription management and monitoring purposes.  This
   includes subscriptions that have been set up via
   RPC primitives as well as subscriptions that have been
   established via configuration.";
list subscription {
  key "id";
  description
    "The identity and specific parameters of a subscription.
     Subscriptions in this list can be created using a control
     channel or RPC or can be established through configuration.

     If the 'kill-subscription' RPC or configuration operations
     are used to delete a subscription, a
     'subscription-terminated' message is sent to any active or
     suspended receivers.";
  leaf id {
    type sn:subscription-id;
    description
      "Identifier of a subscription; unique in a given
       publisher.";
  }
  choice target {
    mandatory true;
    description
      "Identifies the source of information against which a
       subscription is being applied as well as specifics on the
       subset of information desired from that source.";
    case stream {
      description
        "An event stream filter can be applied to a
         subscription. That filter should always be fully
         included in the Data Collection manifest, i.e. common
         filters need to be added here.";
      leaf stream {
        type leafref {
          path "/p-mf:platforms"
             + "/p-mf:platform[p-mf:id=current()/../../.."
             + "/platform-id]/p-mf:yang-push-streams"
             + "/p-mf:stream/p-mf:name";
        }
        mandatory true;
        description
          "Indicates the event stream to be considered for
           this subscription.";
```

```
    }
    uses sn:stream-filter-elements;
  }
  case datastore {
    description
      "Yang-push subscription from a datastore.
       That filter should always be fully included in the
       Data Collection manifest, i.e. common filters need
       to be added here.";
    leaf datastore {
      type leafref {
        path "/p-mf:platforms"
           + "/p-mf:platform[p-mf:id=current()/../../.."
           + "/platform-id]/p-mf:yang-library"
           + "/p-mf:datastore/p-mf:name";
      }
      mandatory true;
      description
        "Datastore from which to retrieve data.";
    }
    uses yp:selection-filter-types;
  }
}
leaf transport {
  type sn:transport;
  description
    "For a configured subscription, this leaf specifies the
     transport used to deliver messages destined for all
     receivers of that subscription.";
}
leaf encoding {
  type sn:encoding;
  description
    "The type of encoding for notification messages.  For a
     dynamic subscription, if not included as part of an
     'establish-subscription' RPC, the encoding will be
     populated with the encoding used by that RPC.  For a
     configured subscription, if not explicitly configured,
     the encoding will be the default encoding for an
     underlying transport.";
}
leaf purpose {
  type string;
  description
    "Open text allowing a configuring entity to embed the
     originator or other specifics of this subscription.";
}
uses sn:update-qos;
uses yp:update-policy-modifiable;
```

```
leaf current-period {
  when '../periodic';
  type yp:centiseconds;
  description
    "Period during two succesive data collections, in the
     current state. Might differ from the configured period
     when the plaftorm might increase the period
     automatically when it is overloaded.";
}
container receivers {
  description
    "Set of receivers in a subscription.";
  list receiver {
    key "name";
    min-elements 1;
    description
      "A host intended as a recipient for the notification
       messages of a subscription.  For configured
       subscriptions, transport-specific network parameters
       (or a leafref to those parameters) may be augmented to
       a specific receiver in this list.";
    leaf name {
      type string;
      description
        "Identifies a unique receiver for a subscription.";
    }
    leaf sent-event-records {
      type yang:zero-based-counter64;
      config false;
      description
        "The number of event records sent to the receiver.
         The count is initialized when a dynamic subscription
         is established or when a configured receiver
         transitions to the 'valid' state.";
    }
    leaf excluded-event-records {
      type yang:zero-based-counter64;
      config false;
      description
        "The number of event records explicitly removed via
         either an event stream filter or an access control
         filter so that they are not passed to a receiver.
         This count is set to zero each time
         'sent-event-records' is initialized.";
    }
    leaf state {
      type enumeration {
        enum active {
          value 1;
```

```
                  description
                    "The receiver is currently being sent any
                     applicable notification messages for the
                     subscription.";
                }
                enum suspended {
                  value 2;
                  description
                    "The receiver state is 'suspended', so the
                     publisher is currently unable to provide
                     notification messages for the subscription.";
                }
                enum connecting {
                  value 3;
                  description
                    "A subscription has been configured, but a
                     'subscription-started' subscription state change
                     notification needs to be successfully received
                     before notification messages are sent. ";
                }
                enum disconnected {
                  value 4;
                  description
                    "A subscription has failed to send a
                     'subscription-started' state change to the
                     receiver.  Additional connection attempts are not
                     currently being made.";
                }
              }
              config false;
              mandatory true;
              description
                "Specifies the state of a subscription from the
                 perspective of a particular receiver.  With this
                 information, it is possible to determine whether a
                 publisher is currently generating notification
                 messages intended for that receiver.";
            }
          }
        }
      }
      // we could augment here with other kind of collection items
    }
  }
}
```

<CODE ENDS>

5. Data Manifest and the Collected Data

5.1. Collecting the Data Manifest

The Data Manifest MUST be streamed and stored along with the collected data. In case the collected data are moved to a different place (typically a database), the Data Manifest MUST follow the collected data. This can render the collected data unusable if that context is lost, for instance when the data is stored without the relevant information. The Data Manifest MUST be updated when the Data Manifest information changes, for example, when a router is upgraded, when a new telemetry subscription is configured, or when the telemetry subscription parameters change. The Data Manifest can itself be considered as a time series, and stored in a similar fashion to the collected data.

The collected data should be mapped to the Data Manifest. Since the Data Manifest will not change as frequently as the collected data itself, it makes sense to map several data to the same Data Manifest. Somehow, the collected data must include a metadata pointing to the corresponding Data Manifest. In case of Data Manifest change, the system should keep the mapping between the data collected so far and the old Data Manifest, and not assume that the latest Data Manifest is valid for the entire time series.

The Platform Manifest is likely to remain the same until the platform is updated. Thus, the Platform Manifest only needs to be collected once per streaming session and updated after a platform reboot.

As this draft specifically focuses on giving context on data collected via streamed telemetry, we can assume that a streaming telemetry system is available. Retrieving the Data Collection Manifest and Platform Manifest can be done either by reusing that streaming telemetry system (in-band) or using another system (out-of-band), for instance by adding headers or saving manifests into a YANG instance file [RFC9195].

We propose to reuse the existing telemetry system (in-band approach) in order to lower the efforts for implementing this draft. To enable a platform supporting streaming telemetry to also support the Data Manifest, it is sufficient that this platform supports the models from Section 3 and Section 4. Recall that each type of manifest has its own rough frequency update, i.e. at reboot for the Platform Manifest and at new subscription or CPU load variation for the Data Collection Manifest. The Data Manifest MUST be streamed with the YANG-Push on-change feature [RFC8641] (also called event-driven telemetry).

5.2. Mapping Collected Data to the Data Manifest

With YANG-push, each notification sent by the device is part of a subscription, which is also one of the YANG keys used to retrieve the Data Manifest, the other key being the platform ID. In order to enable a posteriori retrieval of the Data Manifest associated to a datapoint, the collector must:

- Keep the subscription id and platform id in the metadata of the collected values
- Collect as well the Data Manifest for the subscription associated to the datapoint.

With this information, to retrieve the Data Manifest from the datapoint, the following happens:

- The subscription id and platform id are retrieved from the datapoint metadata
- The Data Manifest for that datapoint is obtained by using the values above as keys.

We don’t focus on the timing aspect as storing both the data and their manifest in a time series database will allow the data scientists to look for the Data Manifest corresponding to the timestamp of the datapoint. In that scenario, the reliability of the collection of the Data Manifest is the same as the reliability of the data collection itself, since the Data Manifest is like any other data.

5.3. Operational Considerations

It is expected that the Data Manifest is streamed directly from the network equipment, along with YANG-Push [RFC8641] data. However, if the network element streaming telemetry does not support yet the YANG modules from the Data Manifest specified in this document, the telemetry collector could populate the Data Manifest from available information collected from the platform. However, this option requires efforts on the telemetry collector side, as the information gathered in the Data Manifest proposed in this document could be scattered among various standard and vendor- specific YANG modules [RFC8199], that depend on the platform.

That Data Manifest should be kept and available even if the source platform is not accessible (from the collection system), or if the platform has been updated (new operating system or new configuration). The Platform Manifest is "pretty" stable and should change only when the platform is updated or patched. On the other hand, the Data Collection Manifest is likely to change each time a

new YANG-Push subscription [RFC8641] is requested and might even change if the platform load increases and collection periods are updated. To separate these two parts, we enclose each of them in its own module.

6. Example

Below is an example of both a Platform manifest and corresponding Data Collection Manifests. The list of YANG modules in the yang-library container is kept empty for brevity.

<CODE BEGINS> file "manifests-example.json"

```
{
  "ietf-platform-manifest:platforms": {
    "platform": [
      {
        "id": "PE1",
        "yang-library": {
          "module-set": [
            {
              "name": "operational"
            }
          ],
          "schema": [
            {
              "name": "operational-schema",
              "module-set": [
                "operational"
              ]
            }
          ],
          "datastore": [
            {
              "name": "ietf-datastores:operational",
              "schema": "operational-schema"
            }
          ]
        }
      }
    ]
  },
  "ietf-data-collection-manifest:data-collections": {
    "data-collection": [
      {
        "platform-id": "PE1",
        "yang-push-subscriptions": {
          "subscription": [
            {
              "id": 4242,
              "datastore": "ietf-datastores:operational",
              "datastore-xpath-filter":
                     "/ietf-interfaces:interfaces/interface/enabled",

              "on-change": {},
              "receivers": {
                "receiver": [
                  {
                    "name": "yp-collector",
                    "state": "active"
                  }
                ]
```

```
            }
          },
          {
            "id": 4243,
            "datastore": "ietf-datastores:operational",
            "datastore-xpath-filter":
      "/ietf-interfaces:interfaces/interface/statistics/in-octets",
            "periodic": {
              "period": 10000
            },
            "current-period": 20000,
            "receivers": {
              "receiver": [
                {
                  "name": "yp-collector",
                  "state": "active"
                }
              ]
            }
          }
        ]
      }
    ]
  }
}
```

<CODE ENDS>

The file above contains the Data Collection Manifest for two Xpaths subscriptions. With the Data Collection Manifest for the first one, with subscription id 4242, the exact semantics of the collected path, here the administrative status of the network interfaces, can be obtained by looking up the module in the yang-library of the corresponding plaftorm manifest, in order to obtain the exact revision of ieft-interfaces used at collection time. Also, the "on-change" container indicates that data will be sent only if there is a change, thus not receiving data indicates that the administrative status of the interface did not change.

The other example of Data Collection Manifest, with subscription id 4243, shows how a periodic subscription is reported. In that example, the current-period indicates that the requested period of 10s (1000 centiseconds) could not be attained and is now of 20s, for instance because the device is overloaded.

7. Security Considerations

As we are reusing an existing telemetry system, the security considerations lies with the new content divulged in the new manifests. Appropriate access control must be associated to the corresponding leafs and containers.

The integrity and provenance of the data of the collection manifest can be ensured by a signing mechanism such as [I-D.lopez-opsawg-yang-provenance].

8. IANA Considerations

This document includes no request to IANA.

9. Contributors

10. Open Issues

*Do we want to handle the absence of values, i.e. add information about missed collection or errors in the collection context ? It could also explain why some values are missing. On the other hand, this might also be out scope.

*Regarding the inclusion of ietf-yang-library in our module, do we want to include as well the changes from ietf-yang-library-revisions? What if other information are present in the yang-library from the platform? Should we use a YANG mount to capture them as well (they would not be captured with our use of the main yang-library grouping). Similarly, the ietf-data-collection-manifest.yang includes many lines of copy-pasting from ietf-yang-push.yang and ietf-subscribed-notifications.yang since we want to include the information from these modules. Reusing groupings is not suitable as some leafrefs are pointing to nodes that are not at the same location in our network level module. Maybe we need to find a solution (deviations + some kind of schema mount?) maybe we can live with similar modules since they have common nodes but different purposes?

*Henk: how does this interact with SBOM effort?

*What is the link with the RFC8345 NodeId and IVY?

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/

RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

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

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

[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>.

[RFC9195] Lengyel, B. and B. Claise, "A File Format for YANG Instance Data", RFC 9195, DOI 10.17487/RFC9195, February 2022, <https://www.rfc-editor.org/info/rfc9195>.

11.2. Informative References

[I-D.clacla-netmod-model-catalog] Clarke, J. and B. Claise, "YANG module for yangcatalog.org", Work in Progress, Internet-Draft, draft-clacla-netmod-model-catalog-03, 3 April 2018, <https://datatracker.ietf.org/doc/html/draft-clacla-netmod-model-catalog-03>.

[I-D.claise-netconf-metadata-for-collection] Claise, B., Nayyar, M., and A. R. Sesani, "Per-Node Capabilities for Optimum Operational Data Collection", Work in Progress, Internet-Draft, draft-claise-netconf-metadata-for-collection-02, 12 July 2021, <https://datatracker.ietf.org/doc/html/draft-claise-netconf-metadata-for-collection-02>.

[I-D.lopez-opsawg-yang-provenance] Lopez, D. and A. Pastor, "Applying COSE Signatures for YANG Data Provenance", Work in Progress, Internet-Draft, draft-lopez-opsawg-yang-provenance-01, 23 October 2023, <https://datatracker.ietf.org/api/v1/doc/document/draft-lopez-opsawg-yang-provenance/>.

[I-D.tgraf-netconf-notif-sequencing] Graf, T., Quilbeuf, J., and A. H. Feng, "Support of Hostname and Sequencing in YANG Notifications", Work in Progress, Internet-Draft, draft-tgraf-netconf-notif-sequencing-02, 6 October 2023, <https://datatracker.ietf.org/doc/html/draft-tgraf-netconf-notif-sequencing-02>.

[RFC8199] Bogdanovic, D., Claise, B., and C. Moberg, "YANG Module Classification", RFC 8199, DOI 10.17487/RFC8199, July 2017, <https://www.rfc-editor.org/info/rfc8199>.

[RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <https://www.rfc-editor.org/info/rfc8343>.

[RFC9196] Lengyel, B., Clemm, A., and B. Claise, "YANG Modules Describing Capabilities for Systems and Datastore Update Notifications", RFC 9196, DOI 10.17487/RFC9196, February 2022, <https://www.rfc-editor.org/info/rfc9196>.

[RFC9371] Baber, A. and P. Hoffman, "Registration Procedures for Private Enterprise Numbers (PENs)", RFC 9371, DOI 10.17487/RFC9371, March 2023, <https://www.rfc-editor.org/info/rfc9371>.

Appendix A. Example of use based on MDT

In this example, the goal is to collect the administrative status and number of received bytes for the interfaces of a fictional ACME device, and store the result in a Influx database. The metrics are collected via YANG-Push, which is configured by specifying their XPaths and when they should be collected (periodically or on-change). More precisely, we want collect "ietf-interfaces:interfaces/interface/enabled" on every change and "ietf-interfaces:interfaces/interface/statistics/in-octets" every 100 milliseconds. The paths here are referring to the YANG module from [RFC8343]. The configuration of YANG push is out of scope for this document. Since they don’t have the same trigger, each of the path must be collected in its own subscription. Figure 3 presents an example for such a collection.

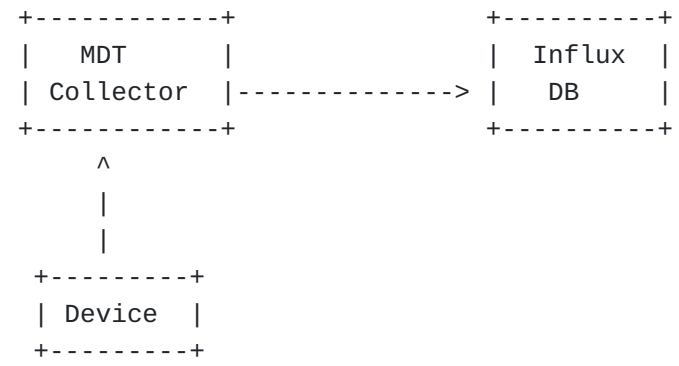

Figure 3: Example of collection from a device to Influx DB

In the scenario from Figure 3, the collector receives YANG-push from the device and stores it into InfluxDB. We first present a version without data manifest and then how to enrich it with the data manifest.

In InfluxDB, a datapoint is specified by giving the name of the measurement, zero or more key value entries named tags, one or more named values called fields and the timestamp for the datapoint. In our case a measurement could be "admin-status". The tags, whose aim to identify a particular instance of the measurement could be the name of the device and the name of the interface. The fields contain the values to store. InfluxDb defines a textual notation, named line protocol, to represent one datapoint per line. We use this line protocol in Figure 4 and Figure 6 to represent the way data could be fed to InfluxDB, omitting the timestamp for readability. See https://docs.influxdata.com/influxdb/cloud/reference/syntax/line-protocol/ for more details.

Without the data manifest, the YANG-push collector is likely to store something similar to Figure 4 in InfluxDB. In that case, only the value is stored, without any way to know how the value was obtained.

```
admin_status,device="PE1",interface="gig1" val=T
sent_bytes,device="PE1",interface="gig1" val=1234
```

Figure 4: Storing datapoints without data manifest

A possibility for keeping the data manifest with the data is to store it directly into InfluxDB. In that case, the collector can subscribe to the data exported by the module presented in this draft and store it inside influxDB. For the Platform Manifest, assuming the platform ID is "PE1", the collector subscribes to the path "ietf-platform-manifest:platforms/platform[id=PE1]". For the Data Collection Manifests, the collector subscribes to the path "ietf-data-collection-manifest:data-collections/data-collection[platform-id="PE1"]/yang-push-subscriptions/subscription[id=X]" where X is the

subscription id of existing subscriptions. The data, for instance serialized in JSON, can be stored in InfluxDB as shown in Figure 5 where "<platform-manifest>" and "<data-manifest>" represent the contents of respectively the Platform Manifest and the Data Collection Manifest.

```
platform-manifest,device="PE1" val=<plaftorm-manifest>
collection-data-manifest,device="PE1",subId=4242 val=<data-manifest>
collection-data-manifest,device="PE1",subId=4243 val=<data-manifest>
```

Figure 5: Storing data manifest

In our example, The link between a collected datapoint and the corresponding Platform Manifest is done via the common "device" tag. In order to link a datapoint with the corresponding Data Collection Manifest, the collector can add fields to specify where the Data Collection Manifest is located for that specific datapoint. For instance, the same datapoints as in Figure 4 could be stored as in Figure 6.

```
admin_status,device="PE1",interface="gig1" val=T,subId=4242
sent_bytes,device="PE1",interface="gig1" val=1234,subId=4243
```

Figure 6: Storing datapoints with data manifest

In our simple example, from the "admin_status" datapoint, one can retrieve the corresponding Platform Manifest by looking at the last value for the "platform-manifest" measurement with the same value for the "device" tag. From the "admin-status" datapoint, one can retrieve the corresponding Data Collection Manifest by looking at the last value for the "data-manifest" measurement with tags "device" and "subId" matching respectively with the tag "device" and the field "subId" of the measurement.

Appendix B. Changes between revisions

v01 -> v02

Updated example with latest version of the model.

v00 (WG adoption) - v01

Solve integrity issue by delegating to [I-D.lopez-opsawg-yang-provenance].

v05 -> v06

Remove YANG packages

- Switch YANG models from device view to network view
- Add PEN number to identify vendors
- Intro rewritten with uses cases
- Added an "Operational Considerations" section
- Switch from MDT to YANG-push

v04 -> v05

- First version of example scenario
- Updated affiliation
- Updated YANG module names to ietf-platform-manifest and ietf-data-collection-manifest
- Unify used terms as defined in the terminology section
- Replaced 'device' with 'platform'
- Split Section 5 into two sections for better readibility

v03 -> v04

- Fix xym error
- Moved terminology after introduction
- Clarified the role of the module

v02 -> v03

- Add when clause in YANG model
- Fix validation errors on YANG modules
- Augment YANG library to handle semantic versioning

v01 -> v02

- Alignment with YANGCatalog YANG module: name, vendor
- Clarify the use of YANG instance file
- Editorial improvements

v00 -> v01

Adding more into data platform: yang packages, whole yanglib module to specify datastores

Setting the right type for periods: int64 -> uint64

Specify the origin datastore for mdt subscription

Set both models to config false

Applying text comments from Mohamed Boucadair

Adding an example of data-manifest file

Adding rationale for reusing telemetry system for collection of the manifests

Export manifest with on change telemetry as opposed to YANG instance file

v00

Initial version

Appendix C. YANG module ietf-yang-push-modif

This section is only here to ensure that the draft passes without compilations error. It should be removed as soon as we can fix the issue.

<CODE BEGINS> file "ietf-yang-push-modif@2023-03-08.yang"

```
module ietf-yang-push-modif {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-yang-push-modif";
  prefix yp;

  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991: Common YANG Data Types";
  }
  import ietf-subscribed-notifications {
    prefix sn;
    reference
      "RFC 8639: Subscription to YANG Notifications";
  }
  import ietf-datastores {
    prefix ds;
    reference
      "RFC 8342: Network Management Datastore Architecture (NMDA)";
  }
  import ietf-restconf {
    prefix rc;
    reference
      "RFC 8040: RESTCONF Protocol";
  }
  import ietf-yang-patch {
    prefix ypatch;
    reference
      "RFC 8072: YANG Patch Media Type";
  }

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

     Author:  Alexander Clemm
              <mailto:ludwig@clemm.org>

     Author:  Eric Voit
              <mailto:evoit@cisco.com>";
  description
    "This module contains YANG specifications for YANG-Push.

     The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL
     NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED',
     'MAY', and 'OPTIONAL' in this document are to be interpreted as
     described in BCP 14 (RFC 2119) (RFC 8174) when, and only when,
```

```
   they appear in all capitals, as shown here.

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

   Redistribution and use in source and binary forms, with or
   without modification, is permitted pursuant to, and subject to
   the license terms contained in, the Simplified BSD License set
   forth in Section 4.c of the IETF Trust's Legal Provisions
   Relating to IETF Documents
   (https://trustee.ietf.org/license-info).

   This version of this YANG module is part of RFC 8641; see the
   RFC itself for full legal notices.";

revision 2023-03-08 {
  description
    "Hack so that ietf-data-collection-manifest compiles.";
  reference
    "RFC 8641: Subscriptions to YANG Datastores";
}
revision 2019-09-09 {
  description
    "Initial revision.";
  reference
    "RFC 8641: Subscriptions to YANG Datastores";
}

/*
 * FEATURES
 */

feature on-change {
  description
    "This feature indicates that on-change triggered subscriptions
     are supported.";
}

/*
 * IDENTITIES
 */
/* Error type identities for datastore subscription */

identity resync-subscription-error {
  description
    "Problem found while attempting to fulfill a
     'resync-subscription' RPC request.";
}

identity cant-exclude {
```

```
  base sn:establish-subscription-error;
  description
    "Unable to remove the set of 'excluded-change' parameters.
     This means that the publisher is unable to restrict
     'push-change-update' notifications to just the change types
     requested for this subscription.";
}

identity datastore-not-subscribable {
  base sn:establish-subscription-error;
  base sn:subscription-terminated-reason;
  description
    "This is not a subscribable datastore.";
}

identity no-such-subscription-resync {
  base resync-subscription-error;
  description
    "The referenced subscription doesn't exist.  This may be as a
     result of a nonexistent subscription ID, an ID that belongs to
     another subscriber, or an ID for a configured subscription.";
}

identity on-change-unsupported {
  base sn:establish-subscription-error;
  description
    "On-change is not supported for any objects that are
     selectable by this filter.";
}

identity on-change-sync-unsupported {
  base sn:establish-subscription-error;
  description
    "Neither 'sync-on-start' nor resynchronization is supported for
     this subscription.  This error will be used for two reasons:
     (1) if an 'establish-subscription' RPC includes
     'sync-on-start' but the publisher can't support sending a
     'push-update' for this subscription for reasons other than
     'on-change-unsupported' or 'sync-too-big'
     (2) if the 'resync-subscription' RPC is invoked for either an
     existing periodic subscription or an on-change subscription
     that can't support resynchronization.";
}

identity period-unsupported {
  base sn:establish-subscription-error;
  base sn:modify-subscription-error;
  base sn:subscription-suspended-reason;
  description
```

```
    "The requested time period or 'dampening-period' is too short.
     This can be for both periodic and on-change subscriptions
     (with or without dampening).  Hints suggesting alternative
     periods may be returned as supplemental information.";
}

identity update-too-big {
  base sn:establish-subscription-error;
  base sn:modify-subscription-error;
  base sn:subscription-suspended-reason;
  description
    "Periodic or on-change push update data trees exceed a maximum
     size limit.  Hints on the estimated size of what was too big
     may be returned as supplemental information.";
}

identity sync-too-big {
  base sn:establish-subscription-error;
  base sn:modify-subscription-error;
  base resync-subscription-error;
  base sn:subscription-suspended-reason;
  description
    "The 'sync-on-start' or resynchronization data tree exceeds a
     maximum size limit.  Hints on the estimated size of what was
     too big may be returned as supplemental information.";
}

identity unchanging-selection {
  base sn:establish-subscription-error;
  base sn:modify-subscription-error;
  base sn:subscription-terminated-reason;
  description
    "The selection filter is unlikely to ever select data tree
     nodes.  This means that based on the subscriber's current
     access rights, the publisher recognizes that the selection
     filter is unlikely to ever select data tree nodes that change.
     Examples for this might be that the node or subtree doesn't
     exist, read access is not permitted for a receiver, or static
     objects that only change at reboot have been chosen.";
}

/*
 * TYPE DEFINITIONS
 */

typedef change-type {
  type enumeration {
    enum create {
      description
```

```
        "A change that refers to the creation of a new
         datastore node.";
    }
    enum delete {
      description
        "A change that refers to the deletion of a
         datastore node.";
    }
    enum insert {
      description
        "A change that refers to the insertion of a new
         user-ordered datastore node.";
    }
    enum move {
      description
        "A change that refers to a reordering of the target
         datastore node.";
    }
    enum replace {
      description
        "A change that refers to a replacement of the target
         datastore node's value.";
    }
  }
  description
    "Specifies different types of datastore changes.

     This type is based on the edit operations defined for
     YANG Patch, with the difference that it is valid for a
     receiver to process an update record that performs a
     'create' operation on a datastore node the receiver believes
     exists or to process a delete on a datastore node the
     receiver believes is missing.";
  reference
    "RFC 8072: YANG Patch Media Type, Section 2.5";
}

typedef selection-filter-ref {
  type leafref {
    path "/sn:filters/yp:selection-filter/yp:filter-id";
  }
  description
    "This type is used to reference a selection filter.";
}

typedef centiseconds {
  type uint32;
  description
    "A period of time, measured in units of 0.01 seconds.";
```

```
}

/*
 * GROUP DEFINITIONS
 */

grouping datastore-criteria {
  description
    "A grouping to define criteria for which selected objects from
     a targeted datastore should be included in push updates.";
  leaf datastore {
    type identityref {
      base ds:datastore;
    }
    mandatory true;
    description
      "Datastore from which to retrieve data.";
  }
  uses selection-filter-objects;
}

grouping selection-filter-types {
  description
    "This grouping defines the types of selectors for objects
     from a datastore.";
  choice datastore-filter-spec {
    description
      "The content filter specification for this request.";
    anydata datastore-subtree-filter {
      if-feature "sn:subtree";
      description
        "This parameter identifies the portions of the
         target datastore to retrieve.";
      reference
        "RFC 6241: Network Configuration Protocol (NETCONF),
                   Section 6";
    }
    leaf datastore-xpath-filter {
      if-feature "sn:xpath";
      type yang:xpath1.0;
      description
        "This parameter contains an XPath expression identifying
         the portions of the target datastore to retrieve.

         If the expression returns a node set, all nodes in the
         node set are selected by the filter.  Otherwise, if the
         expression does not return a node set, the filter
         doesn't select any nodes.
```

```
          The expression is evaluated in the following XPath
          context:

          o  The set of namespace declarations is the set of prefix
             and namespace pairs for all YANG modules implemented
             by the server, where the prefix is the YANG module
             name and the namespace is as defined by the
             'namespace' statement in the YANG module.

             If the leaf is encoded in XML, all namespace
             declarations in scope on the 'stream-xpath-filter'
             leaf element are added to the set of namespace
             declarations.  If a prefix found in the XML is
             already present in the set of namespace declarations,
             the namespace in the XML is used.

          o  The set of variable bindings is empty.

          o  The function library is comprised of the core
             function library and the XPath functions defined in
             Section 10 in RFC 7950.

          o  The context node is the root node of the target
             datastore.";
        reference
          "XML Path Language (XPath) Version 1.0
           (https://www.w3.org/TR/1999/REC-xpath-19991116)
           RFC 7950: The YANG 1.1 Data Modeling Language,
                     Section 10";
      }
    }
  }

  grouping selection-filter-objects {
    description
      "This grouping defines a selector for objects from a
       datastore.";
    choice selection-filter {
      description
        "The source of the selection filter applied to the
         subscription.  This will either (1) come referenced from a
         global list or (2) be provided in the subscription itself.";
      case by-reference {
        description
          "Incorporates a filter that has been configured
           separately.";
        leaf selection-filter-ref {
          type selection-filter-ref;
          mandatory true;
```

```
        description
          "References an existing selection filter that is to be
           applied to the subscription.";
        }
      }
      case within-subscription {
        description
          "A local definition allows a filter to have the same
           lifecycle as the subscription.";
        uses selection-filter-types;
      }
    }
  }

grouping update-policy-modifiable {
  description
    "This grouping describes the datastore-specific subscription
     conditions that can be changed during the lifetime of the
     subscription.";
  choice update-trigger {
    description
      "Defines necessary conditions for sending an event record to
       the subscriber.";
    case periodic {
      container periodic {
        presence "indicates a periodic subscription";
        description
          "The publisher is requested to periodically notify the
           receiver regarding the current values of the datastore
           as defined by the selection filter.";
        leaf period {
          type centiseconds;
          mandatory true;
          description
            "Duration of time that should occur between periodic
             push updates, in units of 0.01 seconds.";
        }
        leaf anchor-time {
          type yang:date-and-time;
          description
            "Designates a timestamp before or after which a series
             of periodic push updates are determined.  The next
             update will take place at a point in time that is a
             multiple of a period from the 'anchor-time'.
             For example, for an 'anchor-time' that is set for the
             top of a particular minute and a period interval of a
             minute, updates will be sent at the top of every
             minute that this subscription is active.";
        }
```

```
      }
    }
    case on-change {
      if-feature "on-change";
      container on-change {
        presence "indicates an on-change subscription";
        description
          "The publisher is requested to notify the receiver
           regarding changes in values in the datastore subset as
           defined by a selection filter.";
        leaf dampening-period {
          type centiseconds;
          default "0";
          description
            "Specifies the minimum interval between the assembly of
             successive update records for a single receiver of a
             subscription.  Whenever subscribed objects change and
             a dampening-period interval (which may be zero) has
             elapsed since the previous update record creation for
             a receiver, any subscribed objects and properties
             that have changed since the previous update record
             will have their current values marshalled and placed
             in a new update record.";
        }
      }
    }
  }
}

grouping update-policy {
  description
    "This grouping describes the datastore-specific subscription
     conditions of a subscription.";
  uses update-policy-modifiable {
    augment "update-trigger/on-change/on-change" {
      description
        "Includes objects that are not modifiable once a
         subscription is established.";
      leaf sync-on-start {
        type boolean;
        default "true";
        description
          "When this object is set to 'false', (1) it restricts an
           on-change subscription from sending 'push-update'
           notifications and (2) pushing a full selection per the
           terms of the selection filter MUST NOT be done for
           this subscription.  Only updates about changes
           (i.e., only 'push-change-update' notifications)
           are sent.  When set to 'true' (the default behavior),
```

```
            in order to facilitate a receiver's synchronization,
            a full update is sent, via a 'push-update' notification,
            when the subscription starts.  After that,
            'push-change-update' notifications are exclusively sent,
            unless the publisher chooses to resync the subscription
            via a new 'push-update' notification.";
        }
        leaf-list excluded-change {
          type change-type;
          description
            "Used to restrict which changes trigger an update.  For
             example, if a 'replace' operation is excluded, only the
             creation and deletion of objects are reported.";
        }
      }
    }
  }

  grouping hints {
    description
      "Parameters associated with an error for a subscription
       made upon a datastore.";
    leaf period-hint {
      type centiseconds;
      description
        "Returned when the requested time period is too short.  This
         hint can assert a viable period for either a periodic push
         cadence or an on-change dampening interval.";
    }
    leaf filter-failure-hint {
      type string;
      description
        "Information describing where and/or why a provided filter
         was unsupportable for a subscription.";
    }
    leaf object-count-estimate {
      type uint32;
      description
        "If there are too many objects that could potentially be
         returned by the selection filter, this identifies the
         estimate of the number of objects that the filter would
         potentially pass.";
    }
    leaf object-count-limit {
      type uint32;
      description
        "If there are too many objects that could be returned by
         the selection filter, this identifies the upper limit of
         the publisher's ability to service this subscription.";
```

```
  }
  leaf kilobytes-estimate {
    type uint32;
    description
      "If the returned information could be beyond the capacity
       of the publisher, this would identify the estimated
       data size that could result from this selection filter.";
  }
  leaf kilobytes-limit {
    type uint32;
    description
      "If the returned information would be beyond the capacity
       of the publisher, this identifies the upper limit of the
       publisher's ability to service this subscription.";
  }
}

/*
 * RPCs
 */

rpc resync-subscription {
  if-feature "on-change";
  description
    "This RPC allows a subscriber of an active on-change
     subscription to request a full push of objects.

     A successful invocation results in a 'push-update' of all
     datastore nodes that the subscriber is permitted to access.
     This RPC can only be invoked on the same session on which the
     subscription is currently active.  In the case of an error, a
     'resync-subscription-error' is sent as part of an error
     response.";
  input {
    leaf id {
      type sn:subscription-id;
      mandatory true;
      description
        "Identifier of the subscription that is to be resynced.";
    }
  }
}

/*
 * NOTIFICATIONS
 */

notification push-update {
  description
```

```
    "This notification contains a push update that in turn contains
     data subscribed to via a subscription.  In the case of a
     periodic subscription, this notification is sent for periodic
     updates.  It can also be used for synchronization updates of
     an on-change subscription.  This notification shall only be
     sent to receivers of a subscription.  It does not constitute
     a general-purpose notification that would be subscribable as
     part of the NETCONF event stream by any receiver.";
  anydata datastore-contents {
    description
      "This contains the updated data.  It constitutes a snapshot
       at the time of update of the set of data that has been
       subscribed to.  The snapshot corresponds to the same
       snapshot that would be returned in a corresponding 'get'
       operation with the same selection filter parameters
       applied.";
  }
  leaf id {
    type sn:subscription-id;
    description
      "This references the subscription that drove the
       notification to be sent.";
  }
  leaf incomplete-update {
    type empty;
    description
      "This is a flag that indicates that not all datastore
       nodes subscribed to are included with this update.  In
       other words, the publisher has failed to fulfill its full
       subscription obligations and, despite its best efforts, is
       providing an incomplete set of objects.";
  }
}

notification push-change-update {
  if-feature "on-change";
  description
    "This notification contains an on-change push update.  This
     notification shall only be sent to the receivers of a
     subscription.  It does not constitute a general-purpose
     notification that would be subscribable as part of the
     NETCONF event stream by any receiver.";
  leaf id {
    type sn:subscription-id;
    description
      "This references the subscription that drove the
       notification to be sent.";
  }
  container datastore-changes {
```

```
    description
      "This contains the set of datastore changes of the target
       datastore, starting at the time of the previous update, per
       the terms of the subscription.";
    uses ypatch:yang-patch;
  }
  leaf incomplete-update {
    type empty;
    description
      "The presence of this object indicates that not all changes
       that have occurred since the last update are included with
       this update.  In other words, the publisher has failed to
       fulfill its full subscription obligations -- for example,
       in cases where it was not able to keep up with a burst of
       changes.";
  }
}

augment "/sn:establish-subscription/sn:input" {
  description
    "This augmentation adds additional subscription parameters
     that apply specifically to datastore updates to RPC input.";
  uses update-policy;
}

augment "/sn:establish-subscription/sn:input/sn:target" {
  description
    "This augmentation adds the datastore as a valid target
     for the subscription to RPC input.";
  case datastore {
    description
      "Information specifying the parameters of a request for a
       datastore subscription.";
    uses datastore-criteria;
  }
}

augment "/sn:modify-subscription/sn:input" {
  description
    "This augmentation adds additional subscription parameters
     specific to datastore updates.";
  uses update-policy-modifiable;
}

augment "/sn:modify-subscription/sn:input/sn:target" {
  description
    "This augmentation adds the datastore as a valid target
     for the subscription to RPC input.";
  case datastore {
```

```
    description
      "Information specifying the parameters of a request for a
       datastore subscription.";
    uses datastore-criteria;
  }
}

augment "/sn:subscription-started" {
  description
    "This augmentation adds datastore-specific objects to
     the notification that a subscription has started.";
  uses update-policy;
}

augment "/sn:subscription-started/sn:target" {
  description
    "This augmentation allows the datastore to be included as
     part of the notification that a subscription has started.";
  case datastore {
    uses datastore-criteria {
      refine "selection-filter/within-subscription" {
        description
          "Specifies the selection filter and where it originated
           from.  If the 'selection-filter-ref' is populated, the
           filter in the subscription came from the 'filters'
           container.  Otherwise, it is populated in-line as part
           of the subscription itself.";
      }
    }
  }
}

augment "/sn:subscription-modified" {
  description
    "This augmentation adds datastore-specific objects to
     the notification that a subscription has been modified.";
  uses update-policy;
}

augment "/sn:subscription-modified/sn:target" {
  description
    "This augmentation allows the datastore to be included as
     part of the notification that a subscription has been
     modified.";
  case datastore {
    uses datastore-criteria {
      refine "selection-filter/within-subscription" {
        description
          "Specifies the selection filter and where it originated
```

```
            from.  If the 'selection-filter-ref' is populated, the
            filter in the subscription came from the 'filters'
            container.  Otherwise, it is populated in-line as part
            of the subscription itself.";
        }
      }
    }
  }

  /*
   * DATA NODES
   */

  augment "/sn:filters" {
    description
      "This augmentation allows the datastore to be included as part
       of the selection-filtering criteria for a subscription.";
    list selection-filter {
      key "filter-id";
      description
        "A list of preconfigured filters that can be applied
         to datastore subscriptions.";
      leaf filter-id {
        type string;
        description
          "An identifier to differentiate between selection
           filters.";
      }
      uses selection-filter-types;
    }
  }

  augment "/sn:subscriptions/sn:subscription" {
    when 'yp:datastore';
    description
      "This augmentation adds objects to a subscription that are
       specific to a datastore subscription, i.e., a subscription to
       a stream of datastore node updates.";
    uses update-policy;
  }

  augment "/sn:subscriptions/sn:subscription/sn:target" {
    description
      "This augmentation allows the datastore to be included as
       part of the selection-filtering criteria for a subscription.";
    case datastore {
      uses datastore-criteria;
    }
  }
```

```
rc:yang-data resync-subscription-error {
  container resync-subscription-error {
    description
      "If a 'resync-subscription' RPC fails, the subscription is
       not resynced and the RPC error response MUST indicate the
       reason for this failure.  This yang-data MAY be inserted as
       structured data in a subscription's RPC error response
       to indicate the reason for the failure.";
    leaf reason {
      type identityref {
        base resync-subscription-error;
      }
      mandatory true;
      description
        "Indicates the reason why the publisher has declined a
         request for subscription resynchronization.";
    }
    uses hints;
  }
}
rc:yang-data establish-subscription-datastore-error-info {
  container establish-subscription-datastore-error-info {
    description
      "If any 'establish-subscription' RPC parameters are
       unsupportable against the datastore, a subscription is not
       created and the RPC error response MUST indicate the reason
       why the subscription failed to be created.  This yang-data
       MAY be inserted as structured data in a subscription's
       RPC error response to indicate the reason for the failure.
       This yang-data MUST be inserted if hints are to be provided
       back to the subscriber.";
    leaf reason {
      type identityref {
        base sn:establish-subscription-error;
      }
      description
        "Indicates the reason why the subscription has failed to
         be created to a targeted datastore.";
    }
    uses hints;
  }
}
rc:yang-data modify-subscription-datastore-error-info {
  container modify-subscription-datastore-error-info {
    description
      "This yang-data MAY be provided as part of a subscription's
       RPC error response when there is a failure of a
       'modify-subscription' RPC that has been made against a
```

```
          datastore.  This yang-data MUST be used if hints are to be
          provided back to the subscriber.";
        leaf reason {
          type identityref {
            base sn:modify-subscription-error;
          }
          description
            "Indicates the reason why the subscription has failed to
             be modified.";
        }
        uses hints;
      }
    }
}
```

<CODE ENDS>

Acknowledgements

Thanks to Mohamed Boucadair and Tianran Zhou for their reviews and comments.

Authors' Addresses

Benoit Claise
Huawei

Email: benoit.claise@huawei.com

Jean Quilbeuf
Huawei

Email: jean.quilbeuf@huawei.com

Diego R. Lopez
Telefonica I+D
Don Ramon de la Cruz, 82
Madrid 28006
Spain

Email: diego.r.lopez@telefonica.com

Ignacio Dominguez
Telefonica I+D
Ronda de la Comunicacion, S/N
Madrid 28050
Spain

Email: ignacio.dominguezmartinez@telefonica.com

Thomas Graf
Swisscom
Binzring 17
CH-8045 Zurich
Switzerland

Email: thomas.graf@swisscom.com