Workgroup: OPSAWG
Internet-Draft:
draft-claise-opsawg-collected-data-manifest-03
Published: 11 July 2022
Intended Status: Standards Track
Expires: 12 January 2023
Authors: B. Claise J. Quilbeuf D. Lopez
Huawei Huawei Telefonica I+D
I. Dominguez T. Graf
Universidad Politecnica de Madrid Swisscom

A Data Manifest for Contextualized Telemetry Data

Abstract

Most network equipment feature telemetry as a mean to monitoring their status. Several protocols exist to this end, for example, the model-driven telemetry governed by YANG models. Some of these protocols provide the data itself, without any contextual information about the collection method. This can render the data unusable if that context is lost, for instance when the data is stored without the relevant information. This document proposes a data manifest, composed of two YANG data models, to store that contextual information along with the collected data, in order to keep the collected data exploitable in the future.

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 12 January 2023.

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.

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 data.

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

2. Introduction

Network elements use Model-driven Telemetry (MDT) to continuously stream information, including both counters and state information. This streamed information is used for network monitoring or closed-loop automation. This streamed data can also be stored in a database (sometimes called a big data lake) for further analysis.

When streaming YANG-structured data with YANG-Push [RFC8641], there is a semantic definition in the corresponding YANG module definition. On top of that definition, it is also important to maintain contextual information about the collection environment.

As an example, a database could store a time series representing the evolution of a specific counter. When analyzing the data, it is important to understand that this counter was requested from the network element at specific cadence, as this exact cadence might not be observed in the time series, potentially implying that the network element was under stress. The same time series might report some values as 0 or might even omit some values. This might be explained by 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 is crucial. Indeed, 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.

Precisely characterizing the source used for producing the data (that is the platform manifest) may also be useful to complete the data collection context. As an example, knowing the exact data source software specification might reveal a particularity in the observed data, explained by a specific bug, or a specific bug fix. This is also necessary to ensure the reliability of the collected data. On top of that, in particular for MDT, it is crucial to know the set of YANG modules supported by the device, 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 must be compiled in a platform manifest.

Some related YANG modules have been specified to retrieve the device capabilities:

* [RFC9196] which models the device 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).

While 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. The goal behind this specification is not to expose new information via YANG objects but rather to define what needs to be kept as metadata (the data manifest) to ensure that the collected data can still be interpreted correctly, even if the source device is not accesible (from the collection system), or if the device has been updated (new operating system or new configuration). This manifest contains two parts, the platform manifest and the data collection manifest. The platform manifest is "pretty" stable and should change only when the device is updated or patched. On the other hand, the data collection manifest is likely to change each time a new MDT subscription is requested and might even change if the device load increases and collection periods are updated. To separate these two parts, we enclose each of them in its own module.

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 collected and how collected data is mapped to the data manifest.

3. Platform Manifest

3.1. Overview of the model

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

```
module: ietf-collected-data-platform-manifest
  +--ro platform
     +--ro name?               string
     +--ro vendor?             string
     +--ro software-version?   string
     +--ro software-flavor?    string
     +--ro os-version?         string
     +--ro os-type?            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
     +--ro packages-set
        +--ro package* [name version]
           +--ro name        -> /pkgs:packages/package/name
           +--ro version     leafref
           +--ro checksum?   leafref
```

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

The platform manifest contains a comprehensive set of information characterize a data source. The platform 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 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 device, 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.

Alternatively, the set of available YANG modules on the device can be described via packages-set which contains a list of references to YANG Packages [I-D.ietf-netmod-yang-packages].

3.2. YANG module ietf-collected-data-platform-manifest

<CODE BEGINS> file "ietf-collected-data-platform-manifest@2021-10-15.yang"

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

  import ietf-yang-library {
    prefix yanglib;
    reference
      "RFC8525: YANG Library";
  }
  import ietf-yang-packages {
    prefix pkgs;
    reference
      "RFC  XXXX: YANG Packages.";
  }
  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>";
  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 2021-10-15 {
  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: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: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: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: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: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 platform {
    config false;
    description
      "Contains information about the platform that allows to identify
       and understand the individual data collection information.  ";
    leaf name {
      type string;
      description
        "Platform on which this module is implemented.";
    }
    leaf vendor {
      type string;
      description
        "Organization that implements that platform.";
    }
    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-library {
        description
          "The YANG library of the device specifying the modules available
           in each of the datastores.";
        uses yanglib:yang-library-parameters;
      }
      container packages-set {
        description
          "Alternatively to module-set, use a list of yang packages to
           describe the list of available schema on the platform";
        uses pkgs:yang-ds-pkg-ref;
      }
    }
}
```

<CODE ENDS>

4. Data Collection Manifest

4.1. Overview of the model

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

```
module: ietf-collected-data-manifest
  +--ro data-collection
     +--ro mdt-subscriptions* [subscription-id]
        +--ro subscription-id           uint64
        +--ro datastore?                ds:datastore-ref
        +--ro mdt-path-data-manifest* [path]
           +--ro path                   yang:xpath1.0
           +--ro requested-period?      uint64
           +--ro actual-period?         uint64
           +--ro on-change?             boolean
           +--ro suppress-redundancy?   boolean
```

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

The data-collection container contains the information related to individual items collection. This subtree currently contains only information about MDT collection. It could be extended and extendable to represent other kinds of data collection.

MDT collection is organized in subscriptions. A given collector can subscribe to one ore more subscriptions that usually contain a list of paths. Such a collector only needs the data manifest for subscriptions it subscribed to. The data manifest for MDT is organized by subscriptions as well so that a collector can select only its subscriptions.

We now have a chicken-and-egg issue if the collector collects the data-manifest via MDT and wants the data-manifest for the data-manifest subscription. First the collector will collect the actual paths that it needs in subscription A. Once it has the subscription id for A, it will need an additional subscription B for the data manifest of paths in A. Then, it would need another subscription C to fetch the data manifest for the subscription B and so on... A possible solution would be adding in the "mdt" container an additional list in that contains the data manifest for every path that is a data manifest. By including that list in subscription B, the collector would have the information about subscription B here.

The "datastore" leaf of the subscription container specifies from which datastore the YANG paths are streamed.

Within a given collection subscription, the granularity of the collection is defined by the path. Note that all devices do not support an arbitrary granularity up to the leaf, usually for performance reasons. Each path currently collected by the device should show up in the mdt-path-data-manifest list.

For each path, the collection context must be specified including:

* 'on-change': when set to true, an update is sent as soon as and only when a value changes. This is also known as Event-Driven Telemetry (EDT). When set to false, the values are sent regularly.

* 'suppress-redundancy' (only when 'on-change' is false): reduce bandwidth usage by sending a regular update only if the value is different from the previous update.

* 'requested-period' (only when 'on-change' is false): period between two updates requested by the client for this path

* 'actual-period' (only when 'on-change 'is false): actual period retained by the platform between two updates. That period could be larger than the requested one as the router can adjust it for performance reasons.

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

4.2. YANG module ietf-collected-data-manifest

<CODE BEGINS> file "ietf-collected-data-manifest@2021-10-15.yang"

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

  import ietf-datastores {
    prefix ds;
    reference
      "RFC 8342: Network Management Datastore Architecture.";
  }
  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>";
  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-collected-data-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 2021-10-15 {
  description
    "Initial revision";
  reference
    "RFC xxxx: Title to be completed";
}

container data-collection {
  config false;
  description
    "Defines the information for each collected object";
  list mdt-subscriptions {
    key "subscription-id";
    description
      "Contains the list of current subscriptions on the
       local device. Enables the collector to select its own
       subscriptions in the list.";
    leaf subscription-id {
      type uint64;
      description
        "Id of the subscription generated by the telemetry emitter.
         The collector can use this id to retrieve information
         about the collection status for the corresponding paths.

         The type is inspired by openconfig-telemetry. TODO check if
         ietf has telemetry modules that we could leafref to.

         path to subscription id in openconfig:
         openconfig-telemetry:telemetry-system/subscriptions/
          persistent-subscriptions/persistent-subscription/state/oid";
    }
    leaf datastore {
      type ds:datastore-ref;
      description
        "The datastore from which the data for this subscription has
         been collected.";
    }
    list mdt-path-data-manifest {
      key "path";
      description
        "Status of the collection for the given path";
      leaf path {
        type yang:xpath1.0;
        description
```

```
            "The XPath context in which this XPath expression is
             evaluated is the datastore selected for the containing
             subscription object. If the datastore is not specified
             then the context is the operational datastore context.";
          }
          leaf requested-period {
            when "../on-change = 'false'";
            type uint64;
            description
              "Requested period, in millisecond, between two successive
               updates.";
          }
          leaf actual-period {
            when "../on-change = 'false'";
            type uint64;
            description
              "Period in milisecond between two successive updates
               actually applied by the plaftorm at configuration time.
               This period can be larger than the requested period as the
               platform might adjust it for performance reasons.";
          }
          leaf on-change {
            type boolean;
            description
              "Whether the path is collected only when there is a
               change, i.e. Event-Driven Telemetry is enabled.";
          }
          leaf suppress-redundancy {
            type boolean;
            description
              "Whether the information is sent at every period or only
               when there is a change between two successive pollings.";
          }
        }
        // we could augment here with other kind of collection items
      }
    }
  }
```

<CODE ENDS>

5. Collecting Data Manifest and Mapping Data to Data Manifest

The data manifest MUST be streamed all with the data 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 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 paremeters change.

The data should be mapped to the data manifest. Since the data manifest will not change as frequently as the 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.

The platform manifest is likely to remain the same until the device is updated. So, the platform manifest only needs to be collected once per streaming session and updated after a device 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. Collecting the data and platform manifests 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 instace 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 data collection manifests, it is sufficient that this device 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).

5.1. Mapping Collected Data to the Data Manifest

With MDT, a particular datapoint is always associated to a path that is itself part of a subscription. In order to enable a posteriori

retrieval of the data manifest associated to a datapoint, the collector must:

- keep the path in the metadata of the collected values
- collect as well the data-manifest for the subscription and path associated to the datapoint.

With this information, to retrieve the data manifest from the datapoint, the following happens:

- the path is retrieved from the datapoint metadata
- the data-manifest for that path is retrieved by looking up on the collected data-manifest.

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. For telemetry based on gRPC for instance, a disconnection to the server would be detected as the HTTP connection would fail.

6. Example

Below is an example of a data-manifest file:

<CODE BEGINS> file "ietf-collected-data-manifest@2021-10-15.yang"

```
{
  "ietf-yang-instance-data:instance-data-set": {
    "name": "data-manifest-example",
    "content-schema": {
      "module": "ietf-collected-data-manifest@2021-10-15"
    },
    "timestamp": "2022-02-24T09:45:03Z",
    "description": [
      "Data manifest for the subscription 4242"
    ],
    "content-data": {
      "ietf-collected-data-manifest:data-collection": {
        "mdt-subscriptions": [
          {
            "subscription-id": 4242,
            "mdt-path-data-manifest": [
              {
                "path": "/ietf-interfaces:interfaces/interface/enabled",
                "requested-period": 100,
                "current-period": 10000,
                "on-change": false,
                "suppress-redundancy": false
              },
              {
                "path": "/ietf-interfaces:interfaces/interface/statistic
                "requested-period": 100,
                "current-period": 100,
                "on-change": false,
                "suppress-redundancy": false
              }
            ]
          }
        ]
      }
    }
  }
}
```

<CODE ENDS>

The file above contains the data manifest for paths collected in the subscription with id 4242. The requested period for both path is this subscription was 100ms, however the status of the interface could only be collected every 10s.

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.

8. IANA Considerations

This document includes no request to IANA.

9. Contributors

10. Open Issues

* Do we want to the hardware specifications, next to the OS information? How to fully characterize a virtual device? Do we need to include the vendor (as PEN for instance https://www.iana.org/assignments/enterprise-numbers/enterprise-numbers) ?

* 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.

* How do we handle other kinds of collection than MDT like netflow, SNMP, CLI ? How do we map the collected data to the data-manifest ?

* Align the terms with the YANG Push specifications. Ex: path to subscription (TBC)

* Better explain the on-change example.

* 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-libary 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).

* Henk: how does this interact with SBOM effort?

* Eliot: important to give integrity of the information a lot of thought. Threat model to be considered.

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>.

[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, <http://www.ietf.org/internet-drafts/draft-clacla-netmod-model-catalog-03.txt>.

[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://www.ietf.org/archive/id/draft-claise-netconf-metadata-for-collection-02.txt>.

[I-D.ietf-netmod-yang-packages] Wilton, R., Rahman, R., Clarke, J., Sterne, J., and B. Wu, "YANG Packages", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-packages-03, 4 March 2022, <https://www.ietf.org/archive/id/draft-ietf-netmod-yang-packages-03.txt>.

[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>.

Appendix A. Changes between revisions

Version 3

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

Version 2

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

Version 1

- 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

Version 0

- Initial version

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
Universidad Politecnica de Madrid
Avenida Complutense 30
Madrid 28040
Spain

Email: i.dominguezm@upm.es

Thomas Graf
Swisscom
Binzring 17
CH-8045 Zurich
Switzerland

Email: thomas.graf@swisscom.com