NMOP
Internet-Draft
Intended status: Standards Track
Expires: 22 July 2026

A. Elhassany
T. Graf
Swisscom
P. Lucente
NTT

18 January 2026

Extensible YANG Model for Network Telemetry Messages
draft-ietf-nmop-message-broker-telemetry-message-04

Abstract

This document defines an extensible message schema in YANG to be used at data collection to transform Network Telemetry messages into external systems such as Message Brokers. The extensible message schema enables data collectors to add metadata for the provenance of the operational network data.

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 22 July 2026.

Copyright Notice

Copyright (c) 2026 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction

Nowadays network operators are using machine and human readable YANG [RFC7950] to model their configurations and obtain YANG modelled operational data from their networks.

Network operators organize their data in a Data Mesh [Deh22] where a Message Broker such as Apache Kafka [Kaf11] or Apache Pulsar [Pul16] facilitates the exchange of messages among data processing components.

Today, subscribing to a YANG datastore, publishing a YANG modeled notifications message from the network and viewing the data in a time series database, manual labor is needed to perform data transformation to make a Message Broker and its data processing components with YANG notifications interoperable.

Even though YANG is intended to ease data management, this promise has not yet been fulfilled for Network Telemetry [RFC9232].

An Architecture for YANG-Push to Message Broker Integration [I-D.ietf-nmop-yang-message-broker-integration] defined an architecture for integrating YANG-Push with Message Brokers for a Data Mesh architecture. How the notification messages at a YANG-Push Receiver is being transformed to the Message Broker is being described in Section Section 4.5 of [I-D.ietf-nmop-yang-message-broker-integration], however the produced message format left unspecified.

The message could be published as it was received from the network to their organization's Message Broker. However, this approach is insufficient for correct human and automated understanding of the data generated by the network. This insufficiency stems from not presenting a holistic picture along with the data generated by the network. In particular, when a data consumer in the data mesh consumes a YANG message from their organization's Message Broker, they cannot answer simple questions such as:

* Which network operating system collected the data?

* To which network platform belongs the network node?

* What is the subscribed xpath, sub-tree filter and its schema reference?

* When did a data collector received the data?

* What additional metadata is necessary for a consumer to make sense of the data?

Section 7.2 of [I-D.ietf-opsawg-collected-data-manifest] describes the content of a Data Manifest and how it is being mapped to the collected Network Telemetry data. The "ietf-telemetry-message" YANG module defined in this document makes use of the platform-details grouping defined in Section 5.2 of [I-D.ietf-opsawg-collected-data-manifest] for the network node and the data collector.

This document defines a standard YANG envelope message to carry with the collected Network Telemetry notifications the provenance and metadata information for a YANG data exchange between between Message Broker producers and consumers as described in Section 4.5 and 4.6 [I-D.ietf-nmop-yang-message-broker-integration]. The described YANG model fascilitates JSON [RFC7951] and CBOR [RFC9254] serialization.

2. Conventions and Definitions

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.

2.1. Terminology

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

The term "Network Telemetry" is used as defined in [RFC9232]. This document uses the term export and collection to distinguish between the data export and collection.

The term "Message Broker" is used as defined in [I-D.ietf-nmop-yang-message-broker-integration].

The term "Data Manifest" is used as defined in [I-D.ietf-opsawg-collected-data-manifest]. The term provenance is used in general to describe the origin, history, and authenticity of an asset which then is described in a manifest.

In addition, this document reuses the terms "Notification Metadata" and "Notification Envelope" defined in [I-D.ietf-netconf-notif-envelope] for the use in Message Broker environment:

Notification Metadata: Additional data describing the context of a notification that is sent in each message, e.g. which node generated the messsage or at which time the notification was published.

Notification Envelope: YANG structure encapsulating the payload of a notification, allowing the inclusion of metadata.

3. YANG Modules

This document defines two YANG modules, an extensible YANG module for Network Telemetry messages defined in Figure 3 and a YANG-Push extension defined in Figure 4.

The extensible YANG module for Network Telemetry messages defines an envelope message schema which adds two provenance and two metadata categories to the collected Network Telemetry data.

The YANG-Push extension adds YANG-Push specific subscription metadata to the Network Telemetry protocol provenance of the envelope.

Network Node Manifest: The OPTIONAL "network-node-manifest" container in Figure 3 contains the Data Manifest about the network node that exported Network Telemetry data. It adds information such as the node name, vendor name, and software and operating system version. Each leaf is OPTIONAL.

Network Telemetry Protocol Metadata: The "telemetry-message-metadata" container in Figure 3 contains the Network Telemetry session information between the collector and the network node. It adds fields such as the Network Telemetry protocol name, the IP addresses and ports of that transport session and the time the event was exported at the network node and received at the data collector. Apart from the collection-timestamp, notification-event, session-protocol and remote-address leafs all other leafs are OPTIONAL. These Network Telemetry session informations allow a network operator to troubleshoot whether the metrics were obtained through NETCONF or RESTCONF <GET> operations or YANG-Push and measure delay and loss related issues. Moreover, this document also defines with Figure 4 an extension specific to YANG-Push that includes YANG-Push subscription information relevant to the Network Telemetry ession information.

Data Collection Manifest: The OPTIONAL "data-collection-manifest" container in Figure 3 contains the Data Manifest of the data collector which collected the Network Telemetry data. The data type is the same as the previous category but specific to the collector node.

Network Operator Metadata: The OPTIONAL "labels" list in the "network-operator-metadata" container in Figure 3 contains the operator specific metadata. Some operators enrich the collected data with specific information. For instance: type of the network node (provider or provider edge node) or which operational unit the node is operated by. For this purpose the document defines a generic metadata map with key/values that can be used freely by the network operator.

Payload: The MUST anydata "payload" contains the Message or notification received from the network element. In case of YANG-Push, the Section 7.16 of YANG [RFC7950] "notification" statement is encoded in the "payload" node. The name and namespace of this payload element are determined by the YANG module containing the "notification" statement representing the notification message.

```
module: ietf-telemetry-message

  structure message:
    +-- network-node-manifest {network-node-manifest}?
    |  +-- name?               string
    |  +-- vendor?             string
    |  +-- vendor-pen?         uint32
    |  +-- software-version?   string
    |  +-- software-flavor?    string
    |  +-- os-version?         string
    |  +-- os-type?            string
    +-- telemetry-message-metadata
    |  +-- node-export-timestamp?   yang:date-and-time
    |  +-- collection-timestamp     yang:date-and-time
    |  +-- notification-event       telemetry-notification-event-type
    |  +-- sequence-number?         yang:counter32
    |  +-- session-protocol         telemetry-session-protocol-type
    |  +-- export-address           inet:host
    |  +-- export-port?             inet:port-number
    |  +-- collection-address?      inet:host
    |  +-- collection-port?         inet:port-number
    +-- data-collection-manifest {data-collection-manifest}?
    |  +-- name?               string
    |  +-- vendor?             string
    |  +-- vendor-pen?         uint32
    |  +-- software-version?   string
    |  +-- software-flavor?    string
    |  +-- os-version?         string
    |  +-- os-type?            string
    +-- network-operator-metadata
    |  +-- labels* [name]
    |     +-- name       string
    |     +-- (value)
    |        +--:(number-choice)
    |        |  +-- (number-choice)?
    |        |     +--:(number-value)
    |        |        +-- number-value?   uint64
    |        +--:(string-choice)
    |        |  +-- (string-choice)?
    |        |     +--:(string-value)
    |        |        +-- string-value?   string
    |        +--:(anydata-choice)
    |           +-- (anydata-choice)?
    |              +--:(anydata-values)
    |                 +-- anydata-values?   anydata
    +-- payload?                     anydata
```

Figure 1: YANG tree diagram for 'ietf-telemetry-message' module.

```
module: ietf-yang-push-telemetry-message

  augment-structure /tm:message/tm:telemetry-message-metadata:
    +-- yang-push-subscription
       +-- id?                          sn:subscription-id
       +-- (filter-spec)?
       |  +--:(subtree-filter)?
       |  |  +-- subtree-filter?   anydata
       |  +--:(xpath-filter)?
       |     +-- xpath-filter?     yang:xpath1.0
       +-- (target)?
       |  +--:(stream)
       |  |  +-- stream?   string
       |  +--:(datastore)
       |     +-- datastore?   identityref
       +-- transport?                   sn:transport
       +-- encoding?                    sn:encoding
       +-- purpose?                     string
       +-- (update-trigger)?
       |  +--:(periodic)
       |  |  +-- periodic!
       |  |     +-- period?        yp:centiseconds
       |  |     +-- anchor-time?   yang:date-and-time
       |  +--:(on-change)
       |     +-- on-change!
       |        +-- dampening-period?   yp:centiseconds
       |        +-- sync-on-start?      boolean
       +-- module* [name]
       |  +-- name        yang:yang-identifier
       |  +-- revision?   rev:revision-date
       |  +-- version?    ysver:version
       +-- yang-library-content-id?   string
```

Figure 2: ietf-yang-push-telemetry-message tree

```
<CODE BEGINS> file "ietf-telemetry-message@2025-10-19.yang"
module ietf-telemetry-message {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-telemetry-message";
  prefix tm;

  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991: Common YANG Data Types";
  }
  import ietf-inet-types {
    prefix inet;
```

```
      reference
        "RFC 6991: Common YANG Data Types";
    }
    import ietf-platform-manifest {
      prefix p-mf;
      reference
        "draft-ietf-opsawg-collected-data-manifest: A Data Manifest for
         Contextualized Telemetry Data";
    }
    import ietf-yang-structure-ext {
      prefix sx;
      reference
        "RFC 8791: YANG Data Structure Extensions";
    }

    organization
      "IETF Draft";
    contact
      "Author:    Ahmed Elhassany
                  <mailto:ahmed.elhassany@swisscom.com>

                  Thomas Graf
                  <mailto:thomas.graf@swisscom.com>";
    description
      "This YANG module defines an extensible message schema to be used at
       data collection to transform Network Telemetry messages towards
       external systems such as Message Brokers.

       Copyright (c) 2025 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 2025-10-19 {
      description
        "Initial revision.";
      reference
        "RFC XXXX";
    }

    sx:structure "message" {
```

```
      description
        "Telemetry message used within the Data Mesh";
      container network-node-manifest {
        if-feature
          "network-node-manifest";
        description
          "Contains the Data Manifest about the network node that
           exported Network Telemetry data.";
        uses p-mf:platform-details;
      }
      container telemetry-message-metadata {
        description
          "contains the session information about the session between the
           collector and the network node and the publishing process
           of the collector.";
        leaf node-export-timestamp {
          type yang:date-and-time;
          description
            "Timestamp when the Network Telemetry data has been exported
             from network element. This can be obtained in YANG-Push from
             event-time defined in draft-ietf-netconf-notif-envelope or
             in IPFIX from the export time in the message header as
             defined in RFC 7011 or in BMP from the timestamp in The
             per-peer header as defined in RFC 7854.";
        }
        leaf collection-timestamp {
          type yang:date-and-time;
          mandatory true;
          description
            "Timestamp when the data collector collected the Network
             Telemetry data from the network element.";
        }
        leaf notification-event {
          type telemetry-notification-event-type;
          mandatory true;
          description
            "Describes wherever the notification was received and forwarded
             from the network node or generated or removed from the data
             collector state cache.";
        }
        leaf sequence-number {
          type yang:counter32;
          description
            "Unique unique sequence number for each published telemetry
             message by the collector. The initial number is 1 and
             counts up by 1 at every published telemetry message
             until it reaches 4294967295 and wraps around.";
        }
```

```
          leaf session-protocol {
            type telemetry-session-protocol-type;
            mandatory true;
            description
              "Session protocol used to collect the Network Telemetry data
               from the network node.";
          }
          leaf export-address {
            type inet:host;
            mandatory true;
            description
              "Network node IP address from where the Network Telemetry data
               was exported from.";
          }
          leaf export-port {
            type inet:port-number;
            description
              "Network node transport port number from where the Network
               Telemetry data was exported.";
          }
          leaf collection-address {
            type inet:host;
            description
              "Data collector IP address at which the Network Telemetry
               data was collected.";
          }
          leaf collection-port {
            type inet:port-number;
            description
              "Data collector transport port number at which the Network
               Telemetry data was collected.";
          }
        }
        container data-collection-manifest {
          if-feature
            "data-collection-manifest";
          description
            "Contains the Data Manifest of the data collector which
             collected the Network Telemetry data.";
          uses p-mf:platform-details;
        }
        container network-operator-metadata {
          description
            "Network operator specific metadata added by the Network
             Telemetry data collection.";
          list labels {
            key
              "name";
```

```
          description
            "Abritrary labels assigned by the data collector.";
          leaf name {
            type string {
              length
                "1..max";
            }
            description
              "Label name.";
          }
          choice value {
            mandatory true;
            description
              "label value";
            choice number-choice {
              description
                "Number value";
              leaf number-value {
                type uint64;
                description
                  "Number value";
              }
            }
            choice string-choice {
              description
                "String value";
              leaf string-value {
                type string;
                description
                  "String value";
              }
            }
            choice anydata-choice {
              description
                "YANG anydata value";
              anydata anydata-values {
                description
                  "anydata yang";
              }
            }
          }
        }
      }
      anydata payload {
        description
          "Message or notification received from network element.";
      }
    }
```

```
feature network-node-manifest {
  description
    "This feature indicates the network node manifest support.";
}

feature data-collection-manifest {
  description
    "This feature indicates the data collection manifest support.";
}

identity session-protocol {
  description
    "Base identity to represent session protocols.";
}

identity yang-push {
  base session-protocol;
  description
    "YANG-Push in RFC 8640 or RFC 8641 or RFC 8650.";
  reference
    "RFC 8640, RFC 8641, RFC 8650: YANG-Push Events and Notifications
     for Datastores.";
}

identity netconf {
  base session-protocol;
  description
    "NETCONF GET RPC as described in RFC 6241.";
  reference
    "RFC 6241: NETCONF GET RPC.";
}

identity restconf {
  base session-protocol;
  description
    "RESTCONF HTTP GET as described in RFC 8040.";
  reference
    "RFC 8040: HTTP GET.";
}

typedef telemetry-notification-event-type {
  type enumeration {
    enum "log" {
      description
        "Collector is reporting the event as it arrived from the
         network element.";
    }
    enum "update" {
```

```
            description
              "Collector has updated an entry inside its local cache.
               This could be triggered by an event from the network for
               instance interface operational status changed or an internal
               event in the collector, such as a timer triggered to referesh
               old enteries.";
          }
          enum "delete" {
            description
              "Collector has deleted an entry from its local cache.";
          }
        }
        description
          "Type of event reported by the collector.";
      }

      typedef telemetry-session-protocol-type {
        type identityref {
          base session-protocol;
        }
        description
          "Network Telemetry protocol used to deliver the notification
           between the network node and the data collector.";
      }
    }
    <CODE ENDS>
```

Figure 3: YANG 'ietf-telemetry-message' module.

```
    <CODE BEGINS> file "ietf-yang-push-telemetry-message@2025-10-19.yang"
    module ietf-yang-push-telemetry-message {
      yang-version 1.1;
      namespace "urn:ietf:params:xml:ns:yang:ietf-yang-push-telemetry-message";
      prefix yptm;

      import ietf-subscribed-notifications {
        prefix sn;
        reference
          "RFC 8639: Subscription to YANG Notifications";
      }
      import ietf-telemetry-message {
        prefix tm;
        reference
          "draft-netana-nmop-message-broker-telemetry-message: Extensible
           YANG Model for Network Telemetry Messages";
      }
      import ietf-yang-push {
        prefix yp;
```

```
  reference
    "RFC 8641: Subscription to YANG Notifications for Datastore
     Updates";
}
import ietf-yang-types {
  prefix yang;
  reference
    "RFC 6991: Common YANG Data Types";
}
import ietf-datastores {
  prefix ds;
  reference
    "RFC 8342: Network Management Datastore Architecture (NMDA)";
}
import ietf-yang-revisions {
  prefix rev;
  reference
    "draft-ietf-netmod-yang-module-versioning: Updated YANG Module
     Revision Handling";
}
import ietf-yang-semver {
  prefix ysver;
  reference
    "draft-ietf-netmod-yang-semver: YANG Semantic Versioning";
}
import ietf-yang-structure-ext {
  prefix sx;
  reference
    "RFC 8791: YANG Data Structure Extensions";
}

organization
  "IETF Draft";
contact
  "Author:    Ahmed Elhassany
              <mailto:ahmed.elhassany@swisscom.com>

              Thomas Graf
              <mailto:thomas.graf@swisscom.com>";
description
  "Adds YANG-Push specific subscription metadata to the data
   collection protocol provenance of the ietf-telemetry-message
   envelope.

   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) 2025 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 2025-10-19 {
      description
        "Initial revision.";
      reference
        "RFC XXXX";
    }

    sx:augment-structure "/tm:message/tm:telemetry-message-metadata" {
      description
        "Augments telemetry-message-metadata with YANG-Push specific
         subscription metadata";
      container yang-push-subscription {
        config false;
        description
          "YANG-Push specific subscription metadata";
        leaf id {
          type sn:subscription-id;
          description
            "This references the affected subscription.";
        }
        choice filter-spec {
          description
            "The content filter specification for this request.";
          anydata subtree-filter {
            description
              "Event stream evaluation criteria or the parameter
               identifies the port of the target datastore encoded in the
               syntax of a subtree filter as defined in RFC 6241,
               Section 6.";
            reference
              "RFC 6241: Network Configuration Protocol (NETCONF),
               Section 6.";
          }
```

```
            leaf xpath-filter {
              type yang:xpath1.0;
              description
                "Event stream evaluation criteria or porting of the target
                 datastore encoded in the syntax of an XPath 1.0
                 expression";
              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";
            }
          }
          choice target {
            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 {
              leaf stream {
                type string;
                description
                  "Indicates the event stream to be considered for this
                   subscription.";
              }
            }
            case datastore {
              leaf datastore {
                type identityref {
                  base ds:datastore;
                }
                description
                  "Datastore from which to retrieve data.";
              }
            }
          }
          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.";
        }
        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 notify periodically the
                 current values of the datastore as defined by the
                 selection filter.";
              leaf period {
                type yp:centiseconds;
                description
                  "Duration of time which should occur between periodic
                   push updates, in one hundredths of a second.";
              }
              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 whole multiple interval
                   from the anchor time.  For example, for an anchor time
                   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 this subscription is active.";
              }
            }
          }
          case on-change {
            container on-change {
              presence
                "indicates an on-change subscription";
              description
                "The publisher is requested to notify changes in values
```

```
                in the datastore subset as defined by a selection
                filter.";
            leaf dampening-period {
              type yp: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, then any subscribed objects and properties
                 which have changed since the previous update record
                 will have their current values marshalled and placed
                 into a new update record.";
            }
            leaf sync-on-start {
              type boolean;
              default
                "true";
              description
                "When this object is set to false, it restricts an
                 on-change subscription from sending push-update
                 notifications.  When false, 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 true (default behavior), in order to facilitate a
                 receiver's synchronization, a full update is sent when
                 the subscription starts using a push-update
                 notification.  After that, push-change-update
                 notifications are exclusively sent unless the publisher
                 chooses to resync the subscription via a new
                 push-update notification.";
            }
          }
        }
      }
      list module {
        key
          "name";
        config false;
        description
          "List of yang-push-module-version grouping defined in
           draft-ietf-netconf-yang-notifications-versioning. The
           revision is not configurable.";
        leaf name {
```

```
            type yang:yang-identifier;
            config false;
            description
              "This references the YANG module name, section 7.1 of
               RFC 7950.";
          }
          leaf revision {
            type rev:revision-date;
            config false;
            description
              "This references the YANG module revision, section 7.1.2
               of RFC 7950, of the sent notification message.";
          }
          leaf version {
            type ysver:version;
            description
              "This references the YANG module semantic version,
               draft-ietf-netmod-yang-semver, of the sent notification
                  \t\t\t\t message.";
          }
        }
        leaf yang-library-content-id {
          type string;
          config false;
          description
            "Contains the YANG library content identifier, RFC 8525,
             information.";
        }
      }
    }
  }
  <CODE ENDS>
```

Figure 4: YANG ’ietf-yang-push-telemetry-message’ module.

4. Use Cases

4.1. Data Chain Troubleshooting

In a distributed system like a data chain described in [I-D.ietf-nmop-yang-message-broker-integration], a single system component could introduce a new issue after a software upgrade of a system component . Thanks to the network node and the data collection manifest, the data engineer is able to identify wherever a software upgrade has been performed on a network node or on a data collector and its impact on the telemetry messages. Such an upgrade could have a potential impact on the YANG schema of the exported YANG-Push Notification which is identifable through the "yang-library-content-

id" and "revision" and "version" defined in [I-D.ietf-netconf-yang-notifications-versioning] and transformed at the data collectors as part of Network Telemetry protocol metadata or the YANG schema id when being produced to the message broker as described in Section 4.5 of [I-D.ietf-nmop-yang-message-broker-integration].

4.2. Data Product Service Level Objective

For Section 3.5 of Service Disruption Detection [I-D.ietf-nmop-network-anomaly-architecture] in Network Anomaly Detection, operational Network Telemetry data needs to to be consumeable as a Section 4.7 of YANG data consumer [I-D.ietf-nmop-yang-message-broker-integration] within the defined time to ensure a holistic view of the network. To measure the delay and loss between the data export on the network node, the data collectors and the YANG data consumer, sequence numbers and timestamps are required. With "hostname" and "sequence-number" defined in Section 3.4 of [I-D.ietf-netconf-notif-envelope] each message is uniquely identifiably and with "event-time" and "observation time" defined in section 3.4 and 3.5 of [I-D.ietf-netconf-notif-envelope] and "collection-time" defined in this document and the timestamp when the data arrived at the YANG data consumer, the loss and delay for a given network node and message can be deducted. Metadata introduced in this document help a data engineer to determine a common denominator and identify the system component causing the delay and loss.

Certain collection protocol types such as polling or push based and subscriptions types, such as on-change or periodical, have different data delay characteristics. Push based on-change subscriptions are expected to export messages almost instantly where polling based subscriptions observe changes depending on their polling interval much later. With the Network Telemetry protocol metadata introduced by this document, based on the update-trigger informations, a YANG data consumer can define for a given YANG-Push subscription id a lower service Level objective than for another. Therefore adjust the detection to be more real-time than without this information.

5. IANA Considerations

This document registers the following two namespace URIs in the IETF XML Registry [RFC3688]:

* URI: urn:ietf:params:xml:ns:yang:ietf-telemetry-message

* Registrant Contact: The IESG.

* XML: N/A; the requested URI is an XML namespace.

* URI: urn:ietf:params:xml:ns:yang:ietf-yang-push-telemetry-message

* Registrant Contact: The IESG.

* XML: N/A; the requested URI is an XML namespace.

This document registers the following two YANG modules in the YANG Module Names registry [RFC3688]:

* Name: ietf-telemetry-message

* Namespace: urn:ietf:params:xml:ns:yang:ietf-telemetry-message

* Prefix: tm

* Reference: RFC XXXX

* Name: ietf-yang-push-telemetry-message

* Namespace: urn:ietf:params:xml:ns:yang:ietf-yang-push-telemetry-message

* Prefix: yptm

* Reference: RFC XXXX

6. Security Considerations

This section is modeled after the template described in Section 3.7 of [I-D.ietf-netmod-rfc8407bis].

The "ietf-telemetry-message" and "ietf-yang-push-telemetry-message" YANG modules defines two data models that are designed to be accessed via YANG-based management protocols, such as NETCONF [RFC6141] and RESTCONF [RFC8040]. These protocols have to use a secure transport layer (e.g., SSH [RFC4252], TLS [RFC8446], and QUIC [RFC9000]) and have to use mutual authentication.

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., "config true", which is the default). All writable data nodes are likely to be reasonably sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) and delete operations to these data nodes without proper protection or authentication can have a negative effect on network operations. The following subtrees and data nodes have particular sensitivities/vulnerabilities:

"There are no particularly sensitive writable data nodes."

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. Specifically, the following subtrees and data nodes have particular sensitivities/ vulnerabilities:

"There are no particularly sensitive readable data nodes."

7. Implementation status

This section provides pointers to existing open source implementations of this draft. Note to the RFC-editor: Please remove this before publishing.

7.1. Netgauze

An open source Network Telemetry data collection implemented "ietf-telemetry-message" and "ietf-yang-push-telemetry-message" .

The open source code can be accessed here: [Netgauze_Github].

Figure 5 provides an example of a JSON encoded, [RFC7951], Network Telemetry message.

```
{
  "ietf-telemetry-message:message": {
    "network-node-manifest": {
      "name": "pe1",
      "vendor": "open source",
      "software-version": "1.1.2"
    },
    "telemetry-message-metadata": {
      "node-export-timestamp": "2024-02-14T12:10:10.10+01:00",
      "collection-timestamp": "2024-02-14T12:10:10.12+01:00",
      "session-protocol": "yang-push",
      "notification-event": "log",
```

```
      "export-address": "192.168.1.100",
      "export-port": 123,
      "collection-address": "192.168.1.1",
      "collection-port": 9991,
      "ietf-yang-push-telemetry-message:yang-push-subscription": {
        "id": 89,
        "xpath-filter": "/ietf-interfaces:interfaces",
        "datastore": "ietf-datastores:operational",
        "transport": "ietf-udp-notif-transport:udp-notif",
        "encoding": "ietf-subscribed-notifications:encode-json",
        "module": [
          {
            "module": "ietf-interfaces",
            "revision": "2018-02-20",
            "version": "2.0.0"
          }
        ],
        "yang-library-content-id": "abc"
      }
    },
    "data-collection-manifest": {
      "name": "collector-1",
      "vendor": "open source",
      "software-version": "2.1.0"
    },
    "network-operator-metadata": {
      "labels": [
        {
          "name": "platform-name",
          "string-value": "name"
        }
      ]
    },
    "payload": {
      "ietf-yang-push:push-update": {
        "id": 89,
        "datastore-contents": {
          "ietf-interfaces:interfaces": {
            "interface": [
              {
                "name": "eth0",
                "oper-status": "down"
              }
            ]
          }
        }
      }
    }
```

```
     }
   }
```

Figure 5: JSON Network Telemetry Example

7.2. Pmacct

An open source Network Telemetry data collection implemented "ietf-telemetry-message" and "ietf-yang-push-telemetry-message" .

The open source code can be accessed here: [Pmacct_Github].

Figure 6 provides an example of a JSON encoded, [RFC7951], Network Telemetry message.

```
========== NOTE: '\' line wrapping per RFC 8792 ===========
{
  "ietf-telemetry-message:message": {
    "data-collection-manifest": {
      "vendor": "pmacct",
      "name": "default",
      "software-version": "1.7.10-git",
      "software-flavor": "20251029-0 (7c22e2cf)"
    },
    "payload": {
      "ietf-yp-notification:envelope": {
        "event-time": "2025-07-18T09:51:00.002Z",
        "hostname": "ipf-zbl1327-r-daisy-91",
        "sequence-number": 1,
        "contents": {
          "ietf-yang-push:push-update": {
            "id": 210,
            "ietf-yp-observation:timestamp": "2025-07-18T09:51:00.002Z",
            "ietf-yp-observation:point-in-time": "current-accounting",
            "datastore-contents": {
              "openconfig-interfaces:interfaces": {
                "interface": [
                  {
                    "name": "TenGigE0/0/0/23",
                    "state": {
                      "counters": {
                        "in-octets": "0",
                        "in-pkts": "0",
                        "in-unicast-pkts": "0",
                        "in-broadcast-pkts": "0",
                        "in-multicast-pkts": "0",
                        "in-discards": "0",
                        "in-errors": "0",
```

```
                        "in-unknown-protos": "0",
                        "in-fcs-errors": "0",
                        "out-octets": "0",
                        "out-pkts": "0",
                        "out-unicast-pkts": "0",
                        "out-broadcast-pkts": "0",
                        "out-multicast-pkts": "0",
                        "out-discards": "0",
                        "out-errors": "0",
                        "Cisco-IOS-XR-openconfig-interfaces-stats-ext:\
                        cisco": {
                          "last-read-time": "1752832251474"
                        }
                      }
                    }
                  }
                ]
              }
            }
          }
        }
      }
    },
    "network-operator-metadata": {
      "labels": [
        {
          "name": "seq",
          "anydata-values": 3
        },
        {
          "name": "serialization",
          "string-value": "json"
        }
      ]
    },
    "telemetry-message-metadata": {
      "notification-event": "log",
      "session-protocol": "yang-push",
      "collection-timestamp": "2025-10-31 11:30:15.324895",
      "export-address": "71.19.157.107",
      "export-port": 31513,
      "ietf-yang-push-telemetry-message:yang-push-subscription": {
        "id": 210,
        "ietf-yang-push:datastore": "ietf-datastores:operational",
        "ietf-yang-push:datastore-xpath-filter": "openconfig-interfaces\
        :interfaces/interface/state/counters",
        "transport": "ietf-udp-notif-transport:udp-notif",
        "ietf-yang-push-revision:module-version": [
```

```
          {
            "module-name": "openconfig-interfaces",
            "revision": "2022-10-25"
          }
        ],
        "ietf-yang-push-revision:yang-library-content-id":\
        "6d13bafd8fd1dc21aa8932d1199b369794881393",
        "encoding": "encode-json",
        "ietf-yang-push:periodic": {
          "period": 3000,
          "anchor-time": "2025-01-01T00:00:30+00:00"
        }
      }
    }
  }
}
```

Figure 6: JSON Network Telemetry Example

8. References

8.1. 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>.

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

[RFC4252] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Authentication Protocol", RFC 4252, DOI 10.17487/RFC4252, January 2006, <https://www.rfc-editor.org/info/rfc4252>.

[RFC6141] Camarillo, G., Ed., Holmberg, C., and Y. Gao, "Re-INVITE and Target-Refresh Request Handling in the Session Initiation Protocol (SIP)", RFC 6141, DOI 10.17487/RFC6141, March 2011, <https://www.rfc-editor.org/info/rfc6141>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

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

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

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[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>.

[RFC8791] Bierman, A., Bj旦rklund, M., and K. Watsen, "YANG Data Structure Extensions", RFC 8791, DOI 10.17487/RFC8791, June 2020, <https://www.rfc-editor.org/info/rfc8791>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/info/rfc9000>.

[RFC9232] Song, H., Qin, F., Martinez-Julia, P., Ciavaglia, L., and A. Wang, "Network Telemetry Framework", RFC 9232, DOI 10.17487/RFC9232, May 2022, <https://www.rfc-editor.org/info/rfc9232>.

[I-D.ietf-nmop-yang-message-broker-integration]
Graf, T. and A. Elhassany, "An Architecture for YANG-Push to Message Broker Integration", Work in Progress, Internet-Draft, draft-ietf-nmop-yang-message-broker-integration-10, 18 January 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-nmop-yang-message-broker-integration-10>.

[I-D.ietf-netconf-notif-envelope]
Feng, A. H., Francois, P., Graf, T., and B. Claise, "Extensible YANG Model for YANG-Push Notifications", Work in Progress, Internet-Draft, draft-ietf-netconf-notif-envelope-03, 20 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-notif-envelope-03>.

[I-D.ietf-opsawg-collected-data-manifest]
Claise, B., Quilbeuf, J., Lopez, D., Martinez-Casanueva, I. D., and T. Graf, "A Data Manifest for Contextualized Telemetry Data", Work in Progress, Internet-Draft, draft-ietf-opsawg-collected-data-manifest-10, 20 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-opsawg-collected-data-manifest-10>.

[I-D.ietf-netmod-yang-module-versioning]
Wilton, R., Rahman, R., Lengyel, B., Clarke, J., and J. Sterne, "Updated YANG Module Revision Handling", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-module-versioning-15, 18 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-yang-module-versioning-15>.

[I-D.ietf-netmod-yang-semver]
Clarke, J., Wilton, R., Rahman, R., Lengyel, B., Sterne, J., and B. Claise, "YANG Semantic Versioning", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-semver-24, 29 September 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-yang-semver-24>.

8.2. Informative References

[RFC3444] Pras, A. and J. Schoenwaelder, "On the Difference between Information Models and Data Models", RFC 3444, DOI 10.17487/RFC3444, January 2003, <https://www.rfc-editor.org/info/rfc3444>.

[RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <https://www.rfc-editor.org/info/rfc7951>.

[RFC9254] Veillette, M., Ed., Petrov, I., Ed., Pelov, A., Bormann, C., and M. Richardson, "Encoding of Data Modeled with YANG in the Concise Binary Object Representation (CBOR)", RFC 9254, DOI 10.17487/RFC9254, July 2022, <https://www.rfc-editor.org/info/rfc9254>.

[I-D.ietf-netmod-rfc8407bis] Bierman, A., Boucadair, M., and Q. Wu, "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", Work in Progress, Internet-Draft, draft-ietf-netmod-rfc8407bis-28, 5 June 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-rfc8407bis-28>.

[I-D.ietf-nmop-network-anomaly-architecture] Graf, T., Du, W., Francois, P., and A. H. Feng, "A Framework for a Network Anomaly Detection Architecture", Work in Progress, Internet-Draft, draft-ietf-nmop-network-anomaly-architecture-06, 21 November 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-nmop-network-anomaly-architecture-06>.

[I-D.ietf-netconf-yang-notifications-versioning] Graf, T., Claise, B., and A. H. Feng, "Support of Versioning in YANG Notifications Subscription", Work in Progress, Internet-Draft, draft-ietf-netconf-yang-notifications-versioning-10, 23 November 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-yang-notifications-versioning-10>.

[Deh22] Dehghani, Z., "Data Mesh", O’Reilly Media, ISBN 9781492092391, March 2022, <https://www.oreilly.com/library/view/data-mesh/9781492092384/>.

[Pul16] Guo, S. and M. Merli, "Apache Pulsar", Apache Software Foundation, January 2016, <https://pulsar.apache.org/>.

[Kaf11] Narkhede, N., "Apache Kafka", Apache Software Foundation, January 2011, <https://kafka.apache.org/>.

[Netgauze_Github] "Netgauze open source Network Telemetry Data Collection", <https://github.com/NetGauze/NetGauze/pull/213>.

[Pmacct_Github]
"Paolo Lucente, Pmacct open source Network Telemetry Data Collection", <https://github.com/pmacct/pmacct>.

Acknowledgements

The authors would like to thank Rob Wilton, Alex Huang Feng, Benoit Claise, Leonardo Rodoni, Reshad Rahman, Dan Voyer, Paolo Lucente and Martin Bj旦rklund for their review and valuable comments.

Authors' Addresses

Ahmed Elhassany
Swisscom
Binzring 17
CH- Zuerich 8045
Switzerland
Email: ahmed.elhassany@swisscom.com

Thomas Graf
Swisscom
Binzring 17
CH- Zuerich 8045
Switzerland
Email: thomas.graf@swisscom.com

Paolo Lucente
NTT
Veemweg 23
Barneveld 3771
Netherlands
Email: paolo@ntt.net