A Data Manifest for Contextualized Telemetry Data

draft-ietf-opsawg-collected-data-manifest-01
B. Claise (Huawei), J. Quilbeuf (Huawei), D. Lopez (Telefonica), I. Dominguez (Telefonica), T. Graf (Swisscom)

IETF 117, OPSAWG
Goal & Problem Statement

• Goal is not to expose new information via YANG but rather to define what needs to be kept as metadata (or Data Manifest) to ensure that the data can still be interpreted correctly even:
  – if the source device is not accessible (from the collection system)
  – If the source device has been updated or has a new configuration

• End goal: analyze the data, from the data collection system, with the proper context, for anomaly detection and, in the end, closed loop automation

• Per-node capability discovery exists
  – YANG Modules describing Capabilities for Systems and Datastore Update Notifications, RFC9196 + YANG Instance Data File Format, RFC9195
  – Per-Node Capabilities for Optimum Operational Data Collection, draft-claise-netconf-metadata-forcollection-03

• But how were data actually metered, under which circumstances?
Data Collection Vantage Point

End goal: analyze the data, from the data collection system, with the proper context, for anomaly detection and, in the end, closed loop automation.
Proposal: Data Manifest

• Data Manifest composed of 2 YANG models for storing the context:
  – **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 was metered.

• “MUST be streamed all with the data and stored along with the collected data.”

• “In case the data are moved to different place (typically a database), the data manifest MUST follow the collected data.”
Changes

• Draft adopted by OPSAWG
• Reference to new draft for ensuring integrity of the data manifest

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].

• Adding the platform-id solves the issue of multiple virtual devices on the same platform: each should have their own ID
Main issue: reuse YANG device module at the network level

- We want to reuse the following modules:
  - ietf-yang-library (augmented by ietf-yang-library-revisions)
  - ietf-subscribed-notifications (augmented by ietf-yang-push)

- These modules are designed to be device modules, we want to use them as network-level module
  - Augments of yang-library have to be rewritten:
    - Revisions augment for yang-library is originally augmenting
      - /yl:yang-library/...
    - But we need to augment
      - /p-mf:plaftorms/p-mf:platform/p-mf:yang-library/...
  - Subscribed-notifications and yang-push do not provide top-level grouping, we have to copy-paste the part that is not in the grouping
  - Yang-push augments a choice in subscribed-notifications with a leaf name that already exists there.
    - OK for yang push because the namespaces are different.
    - Not OK for us, because “uses grouping” result in nodes having the namespace as the “using” modules which is the data-manifest module in our case.
Main issue: reuse YANG device module at the network level (2)

- These issues have already been identified in RFC8528 (YANG Schema Mount), which proposes 3 ways of mounting modules:

1. Design time: The mounted schema is defined along with the mount point in the parent YANG module. In this case, the mounted schema has to be the same for every implementation of the parent module.

2. Implementation time: The mounted schema is defined by a server implementor and is as stable as the YANG library information of the server.

3. Run time: The mounted schema is defined by instance data that is part of the mounted data model. If there are multiple instances of the same mount point (e.g., in multiple entries of a list), the mounted data model may be different for each instance.
Main issue: reuse YANG device module at the network level (3)

Possible solution: define a **Static** YANG Mount extension (new draft) that enables mounting imported modules.

Example:

```yang
module ietf-platform-manifest {
  ...
  import ietf-yang-library { prefix yanglib; }
  import ietf-yang-library-revisions { prefix yl-rev; }
  import ietf-static-schema-mount { prefix static; } // the new extension
  ...
  container platforms{
    ...
    list platform {
      key id;
      leaf id {...}
      ...
      static:mount "yanglib"; // include yang-library top nodes (and RPCs)
      static:mount "yl-rev"; // include augmentations to yang-library for supporting revisions
    }
  }
}
```

The set of static:mount belonging to the same container/list provides an independent YANG context. Data within that context must be valid w.r.t. to the set of modules. The namespaces of the original modules are kept.
Open Questions

• Should we start working on the “Static YANG Mount” draft?
  – Anybody interested to contribute?
  – Is there a better alternative?
• Are we missing anything in the current form of the data manifest?

Feedback, suggestions, issues, PRs:
https://github.com/JeanQuilbeufHuawei/draft-collected-data-manifest
BACKUP SLIDES
Changes: Introduction

• Rewrote and clarified
• Introduces 3 identified use cases:
  – Network Analytics
  – New device onboarding
  – Data mesh principles in networking

• New “Operation Considerations” section for clarifying role of the current draft:
  – Storing data outside of the device
  – Transition period: manifest populated by collector while device support is built
Changes: Platform Manifest

- Network view: top-level container is now a list of ‘platform’ indexed by an `id`.
  - Clarify that we represent data for the network, even if we reuse device modules
  - Platform id left open, suggestion: the ‘sysname’ from draft-tgraf-netconf-notif-sequencing
- Alternative way of specifying vendor: `vendor-pen` from IANA PEN
- Removed dependencies on yang-packages causing YANG errors
- Added list of streams (from RFC8639) supported by the platform for YANG push
Changes: Data Collection Manifest

- Network view: as for platform id
  - Leaf refs to platform:
    - platform-id
    - stream
    - datastore
- Switch to YANG-push instead of MDT
  - Reuse modelling from RFC8639 and RFC8641
  - Index by subscription-id is sufficient (index by path not needed anymore)
- Additional field: ‘current-period’ between two updates, to indicate when the collection period is larger than requested (device overload)
New: renaming

- ietf-collected-data-platform-manifest
  -> ietf-platform-manifest
- ietf-collected-data-manifest
  -> ietf-data-collection-manifest
- Prefixes unchanged
New: example with InfluxDB

Influx input – no manifest:

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

Influx input – with manifest:

```
admin_status,device="PE1",interface="gigi" subId=42,path=<path> val=T
sent_bytes,device="PE1",interface="gigi" subId=42,path=<path> val=1234
```

Possible way of storing manifest in InfluxDB:

```
platform-manifest,device="PE1" val=<platform-manifest>
collection-data-manifest,device="PE1",subId=42 val=<data-manifest>
```

Identifies the Platform Manifest  Identifies the Data Collection Manifest
Open Questions

• What to include for data manifest
  
  https://github.com/JeanQuilbeufHuawei/draft-collected-data-manifest/issues/9
  
  – MDT subscription is a collection of XPaths but NETCONF subscription is a xml filter. What to use as key for subscription contents?
  – Should we include encoding? (XML, JSON, CBOR, Protobuf)

• Identified Improvements:
  – Use Private Enterprise Number (PEN) from IANA to identify vendors

• Open Questions:
  – Do we want to handle data manifest for non-telemetry values? SNMP, IPFIX ...?
  – How to properly specify devices / virtual devices as source?
  – Handle mis-collections? More Counters?
Platform Manifest

New: Include full yang-library, to have datastores

New: rw->ro for every node, Manifest is NOT configurable

New: Add packages-set

New: Vendor
Data Collection Manifest

New list indexed by subscription ID

Specify datastore per subscription
- monitor diff between config and operational
- monitor candidate

MDT path typed as XPath

Unsigned ints for periods
Update Frequency

• Platform manifest:
  – **Update when**: platform changes (i.e. at reboot)

• Data collection manifest:
  – **Update when**: collection condition changes:
    – New subscription
    – Collection period is adjusted based on CPU availability
Collecting Manifests: Use Telemetry

• Platform and data collection manifests are about telemetry
  — we can assume a telemetry system is available
• Event-driven Telemetry/onChange well suited to stream the manifests updates only
• Collectors can choose or not to collect the manifests / header approach would force collector to have the manifests
• Collectors know IDs of their subscription -> YANG key for selecting the data-manifest they need
Mapping Data to Data Manifest

• Collected data needs the following metadata:
  – Source device -> for mapping to platform manifest
  – Subscription ID
  – MDT path

  For mapping data to data manifest

Specifying how to store metadata for collected data is out-of-scope of this draft.