

OPSAWG
Internet-Draft
Intended status: Standards Track
Expires: 21 September 2022

B. Claise
J. Quilbeuf
Huawei
D. Lopez
Telefonica I+D
I. Dominguez
Universidad Politecnica de Madrid
T. Graf
Swisscom
20 March 2022

A Data Manifest for Contextualized Telemetry Data
draft-claise-opsawg-collected-data-manifest-02

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 21 September 2022.

Copyright Notice

Copyright (c) 2022 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. Terminology	2
2. Introduction	3
3. Platform Manifest	4
3.1. Overview of the model	4
3.2. YANG module ietf-collected-data-platform-manifest	6
4. Data Collection Manifest	9
4.1. Overview of the model	9
4.2. YANG module ietf-collected-data-manifest	10
5. Collecting Data Manifest and Mapping Data to Data Manifest	13
5.1. Mapping Collected Data to the Data Manifest	14
6. Example	15
7. Security Considerations	16
8. IANA Considerations	16
9. Contributors	16
10. Open Issues	16
11. References	17
11.1. Normative References	17
11.2. Informative References	17
Appendix A. Changes between revisions	18
Acknowledgements	19
Authors' Addresses	19

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 accessible (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 feature*      yang:yang-identifier
        +--ro deviation*    -> ../../module/name
      +--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 schema* [name]
      +--ro name          string
      +--ro module-set*   -> ../../module-set/name
    +--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       -> /pkgs:packages/package[pkgs:name = current()/../name]/version
      +--ro checksum?     -> /pkgs:packages/package[pkgs:name = current()/../name][pkgs:version = current()/../version]/pkgs:checksum

```

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.claccla-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 platform-manifest;

  import ietf-yang-library {
    prefix yanglib;
    reference
      "RFC8525: YANG Library";
  }
  import ietf-yang-packages {
    prefix pkgs;
    reference
      "RFC XXXX: YANG Packages.";
  }

  organization
    "IETF NETCONF (Network Configuration) Working Group";
  contact
    "WG Web:  <https://datatracker.ietf.org/wg/netconf/>
    WG List:  <mailto:netconf@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 Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>). This version of this YANG module is part of RFC XXXX; see the RFC itself for full legal notices. ";

revision 2021-10-15 {

 description

 "Initial revision";

 reference

 "RFC xxxx: Title to be completed";

}

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 or 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;
}

organization
  "IETF NETCONF (Network Configuration) Working Group";
contact
  "WG Web:    <https://datatracker.ietf.org/wg/netconf/>
  WG List:    <mailto:netconf@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 Simplified BSD License
  set forth in Section 4.c of the IETF Trust's Legal Provisions
  Relating to IETF Documents
  (https://trustee.ietf.org/license-info).
  This version of this YANG module is part of RFC XXXX; 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 {
      type uint64;
      description
        "Requested period, in millisecond, between two successive
        updates.";
      // when on-change is false;
    }
  }
}
```

```
    leaf actual-period {
        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.";
        // when on-change is false;
    }
    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 instance file [RFC9195].

We propose to reuse the existing telemetry system (in-band approach) in order to lower the efforts for implementing this draft. To enable a platform supporting streaming telemetry to also support 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/statistics/in-octets",
                "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-library from the platform? Should we use a YANG mount to capture them as well (they would not be captured with our use of the main yang-library grouping).
- * 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 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

OPSAWG
Internet-Draft
Intended status: Standards Track
Expires: 25 April 2022

O. Gonzalez de Dios
S. Barguil
Telefonica
Q. Wu
Huawei
M. Boucadair
Orange
V. Lopez
Nokia
22 October 2021

A Network YANG Model for Service Attachment Points
draft-dbwb-opsawg-sap-00

Abstract

This document defines a YANG data model for representing an abstract view of the provider network topology containing the points from which its services can be attached (e.g., basic connectivity, VPN, network slices). The data model augments the 'ietf-network' data model by adding the concept of service attachment points (SAPs). The service attachment points are the points to which network services (such as L3VPN or L2VPN) can be attached. The customer endpoint of an attachment circuits are not covered in the SAP network topology.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 25 April 2022.

Copyright Notice

Copyright (c) 2021 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 Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology	3
3. SAP Network Model Usage	4
4. SAP Module Tree Structure	7
5. Relation with other Models	9
6. SAP YANG Module	9
7. IANA Considerations	15
8. Security Considerations	15
9. Acknowledgements	16
10. References	16
10.1. Normative References	16
10.2. Informative References	18
Authors' Addresses	20

1. Introduction

The service attachment point (SAP) is an important architectural concept in many implementations and deployments of services such as VPNs, SDWAN, or managed VoIP services. It has already been used to decide where to attach and, thus, deliver the service in the L3SM [RFC8299] and the L2SM [RFC8466].

This document defines a YANG network model for representing, managing, and controlling the service attachment points (SAPs). The data model augments the 'ietf-network' module [RFC8345] by adding the concept of service attachment points. The service attachment points are abstraction of the points where network services such as L3VPNs or L2VPNs can be attached.

This document does not make any assumption about the service provided by the network to the users. VPN services are used for illustration purposes. This concept can also be used to decide network slice SAPs [I-D.ietf-teas-ietf-network-slices].

In the context of Software-Defined Networking (SDN) [RFC7149][RFC7426], the defined YANG data model in this document can be used to exchange information between control elements, so as to

support VPN service provision and resource management discussed in [I-D.ietf-opsawg-l3sm-l3nm][I-D.ietf-opsawg-l2nm]. Through this data model, the service orchestration layer can learn the available endpoints (i.e., SAPs) of interconnection resource of the underlying network.

The service orchestration layer can determine which endpoint of interconnection to add to L2VPN or L3VPN service. With the help of other data models (e.g., L3SM [RFC8299] or L2SM [RFC8466]), hierarchical control elements could determine the feasibility of an end-to-end IP connectivity or L2VPN connectivity and therefore derive the sequence of domains and the points of interconnection to use.

This document explains the scope and purpose of a SAP network model and its relation with the service models and describes how it can be used by a network operator. The document also shows how the topology and service models fit together.

The YANG data model in this document conforms to the Network Management Datastore Architecture (NMDA) [RFC8342].

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This document assumes that the reader is familiar with the contents of [RFC6241], [RFC7950], and [RFC8309]. The document uses terms from those documents.

Tree diagrams used in this document follow the notation defined in [RFC8340].

This document uses the term "network model" defined in Section 2.1 of [RFC8969].

This document uses the following terms:

Service Provider (SP): The organization responsible for operating the network that offers a service (e.g., a VPN) to customers.

Customer Edge (CE): An equipment that is dedicated to a particular customer and is directly connected to one or more Provider Edges (PEs) via attachment circuits (ACs). A CE is usually located at the customer premises. A CE may be dedicated to a single service

(e.g., L3VPN), although it may support multiple VPNs if each one has separate attachment circuits. A CE can be a router, bridge, switch, etc.

Provider Edge (PE): An equipment owned and managed by the SP that can support multiple services (e.g., VPNs) for different customers. A PE is directly connected to one or more CEs via attachment circuits. A PE is usually located at an SP point of presence (PoP).

Attachment point (AP): Describes a service's end point characteristics and its reference to a Termination Point (TP) of the PE; used as service access point for service.

3. SAP Network Model Usage

Management operations of a service provider network can be automated using a variety of means such as interfaces based on YANG modules [RFC8969]. From that standpoint, and considering the architecture depicted in Figure 1, the goal of this document is to provide a mechanism to show via a YANG-based interface an abstracted network view from the network controller to the service orchestration layer with a focus on where a service can be delivered to customers.

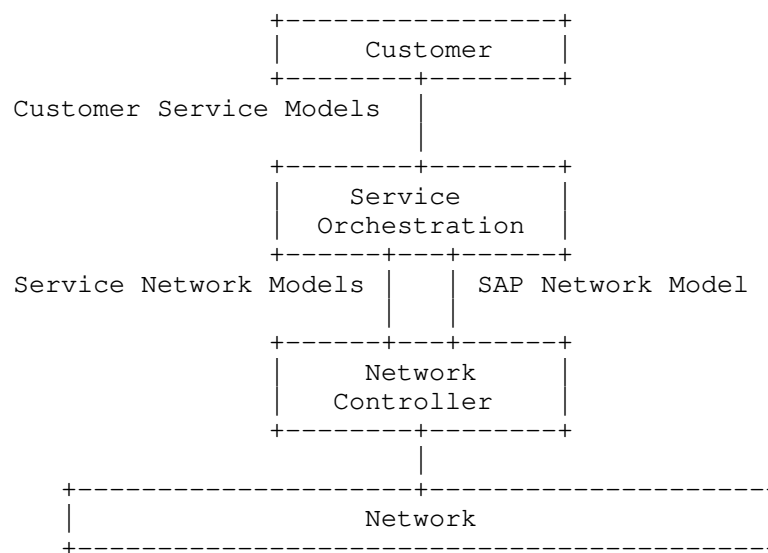


Figure 1: SAP Network Model Usage

Let us consider the example of a typical service provider network (Figure 2), with PE and P nodes.

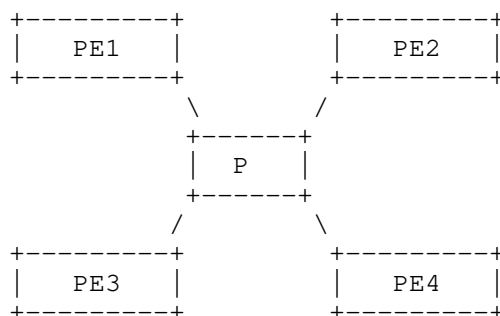


Figure 2: Sample Network Topology

The Service Orchestration layer does not need to know about the internals of the underlying network (e.g., P nodes). Figure 3 shows the abstract network view as seen by the Service Orchestrator. However, this view is not enough to provide to the Service Orchestration layer the information to create services in the network. The service topology need is to be able to expose the set of nodes and the attachment points associated with the nodes from which network services can be grafted (delivered).

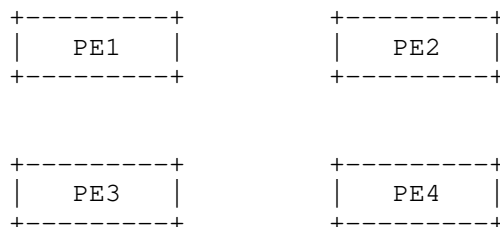


Figure 3: Abstract Network Topology

The Service Orchestration layer would see a set of PEs and a set of client-facing interfaces (physical or logical) to which CEs can be connected (or are actually connected). The Service Orchestration layer can use them to setup the requested services or to commit the delivery of a service. Figure 4 depicts the SAP network topology that is maintained by the network controller and exposed to the Service Orchestration.

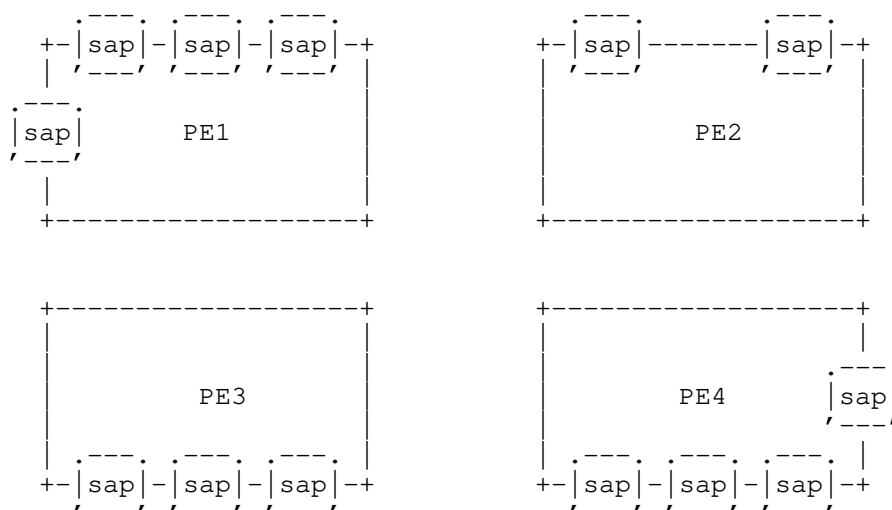


Figure 4: SAP Network Topology

A single SAP network topology can be used for one of multiple service types (e.g., L3VPN, EVPN). The network controller can then expose the service type(s) and associated interfaces via the SAPs.

As shown in Figure 5, the Service Orchestration layer will have also access to a set of Customer Service Model, e.g., an L3SM or L2SM data model in the customer-facing interface and a set of network models, e.g., L3NM and Network topology data models in the resource-facing interface. In this use case, it is assumed that the network controller is unaware of what happens beyond the PEs towards the CEs; it is only responsible for the management and control of the network between PEs.

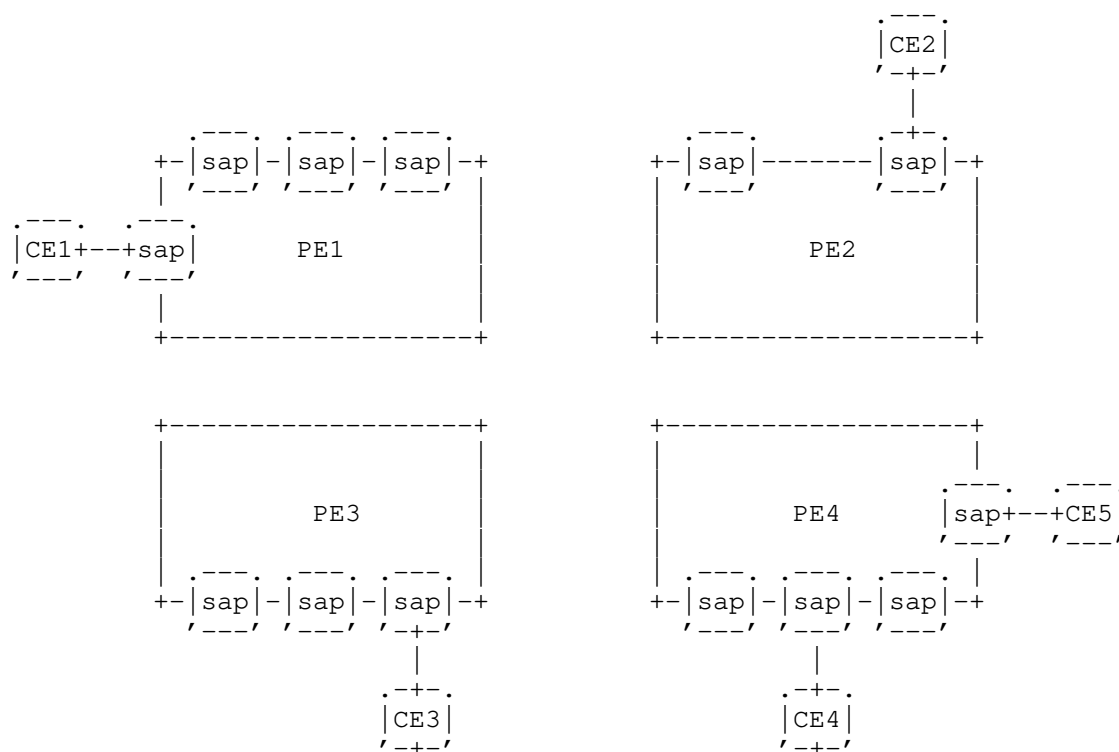


Figure 5: Network Topology with CEs and ACs

4. SAP Module Tree Structure

The SAP network model builds on the network data model defined in the 'ietf-network' module [RFC8345], augmenting the nodes with service attachment points, which anchor the links and are contained in nodes. The 'service-attachment-point' attribute defined in the SAP network model is not a tunnel termination point (TTP) nor a link, but an abstraction of the termination point defined in [RFC8345].

The structure of the 'ietf-sap-ntw' module is shown in Figure 6.

```
module: ietf-sap-ntw
  augment /nw:networks/nw:network/nw:network-types:
    +--rw sap-network!
      +--rw sap-type* identityref
  augment /nw:networks/nw:network/nw:node:
    +--rw service-attachment-point* [attachment-id]
      +--rw attachment-id nt:tp-id
      +--ro interface-type? identityref
      +--rw admin-status? boolean
      +--rw oper-status? boolean
      +--rw encapsulation-type? identityref
      +--rw sap-type* identityref
      +--rw service-description? string
```

Figure 6: YANG Module Structure

A SAP network topology can be used for one single service type or multiple types ("sap-type"). When a SAP topology is used for many service types, the underlying nodes must support at least one of these service types. Examples of supported service types are listed below:

- * L3VPN,
- * Virtual Private LAN Service (VPLS) using BGP [RFC4761],
- * VPLS using Label Distribution Protocol (LDP) [RFC4762],
- * Virtual Private Wire Service (VPWS) [RFC8214],
- * BGP MPLS-Based Ethernet VPN [RFC7432],
- * Ethernet VPN (EVPN) [RFC8365],
- * Provider Backbone Bridging Combined with Ethernet VPN (PBB-EVPN) [RFC7623],
- * Virtual Networks [RFC8453],
- * Enhanced VPN (VPN+) [I-D.ietf-teas-enhanced-vpn], and
- * Network slice [I-D.ietf-teas-ietf-network-slices].

A service attachment point is identified by an interface name ("attachment-id"), an interface type ("type"), a status ("admin-status", and "oper-status"), an encapsulation type ("encapsulation-type"), one or a list of service types ("sap-type") such as L3VPN or network slice, a description of the service(s) ("service-description").

5. Relation with other Models

The SAP network model can be seen as an inventory data associated with service attachment points. The model maintains an inventory of nodes contained in a network based on [RFC8345].

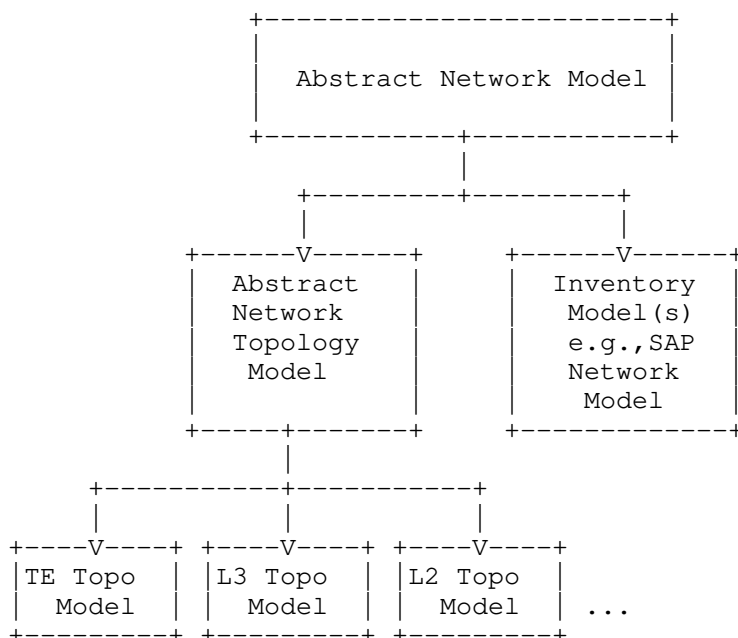


Figure 7: Relation of SAP Network Model to Other Models

Figure 7 depicts the relationship of the SAP network model to other models. The SAP network model augments from the Network model [RFC8345] and imports Network Topology model, while other technology-specific topology models (e.g., TE Topologies model [RFC8795] or L3 Topology model [RFC8346]) augment from the Network Topology.

6. SAP YANG Module

This module imports types from [RFC8343], [RFC8345], and [I-D.ietf-opsawg-vpn-common].

```
<CODE BEGINS> file "ietf-sap-ntw@2021-10-16.yang"
module ietf-sap-ntw {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-sap-ntw";
  prefix sap;

  import ietf-interfaces {
    prefix if;
    reference
      "RFC 8343: A YANG Data Model for Interface Management";
  }
  import ietf-network-topology {
    prefix nt;
    reference
      "RFC 8345: A YANG Data Model for Network
        Topologies, Section 6.2";
  }
  import ietf-network {
    prefix nw;
    reference
      "RFC 8345: A YANG Data Model for Network
        Topologies, Section 6.1";
  }
  import ietf-vpn-common {
    prefix vpn-common;
    reference
      "RFC UUUU: A Layer 2/3 VPN Common YANG Model";
  }

  organization
    "IETF OPSA (Operations and Management Area) Working Group ";
  contact
    "Editor:  Oscar Gonzalez de Dios
      <mailto:oscar.gonzalezdedios@telefonica.com>
     Editor:  Samier Barguil
      <mailto:samier.barguilgiraldo.ext@telefonica.com>
     Editor:  Qin Wu
      <mailto:bill.wu@huawei.com>
     Editor:  Mohamed Boucadair
      <mailto:mohamed.boucadair@orange.com>";
  description
    "This YANG module defines a model for representing, managing,
    and controlling the Service Attachment Points (SAPs) in the
    network topology.

    Copyright (c) 2021 IETF Trust and the persons identified as
    authors of the code.  All rights reserved."
```

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

This version of this YANG module is part of RFC XXXX (<https://www.rfc-editor.org/info/rfcXXXX>); see the RFC itself for full legal notices.";

```
revision 2021-10-16 {
  description
    "Initial version";
  reference
    "RFC XXXX: A Network YANG Model for Service Attachment
      Point (SAP)";
}

identity service-type {
  description
    "Base identity for the service type.";
}

identity l3vpn {
  base service-type;
  description
    "L3VPN service.";
  reference
    "RFC 4364: BGP/MPLS IP Virtual Private Networks (VPNs)";
}

identity enhanced-vpn {
  base service-type;
  description
    "Enhanced VPN (VPN+). VPN+ is an approach that is
      based on existing VPN and Traffic Engineering (TE)
      technologies but adds characteristics that specific
      services require over and above traditional VPNs.";
  reference
    "I-D.ietf-teas-enhanced-vpn:
      A Framework for Enhanced Virtual Private Network
      (VPN+) Services";
}

identity network-slice {
  base service-type;
  description
```

```
    "IETF network slice. An IETF network slice
    is a logical network topology connecting a number of
    endpoints using a set of shared or dedicated network
    resources that are used to satisfy specific service
    objectives.";
  reference
    "I-D.ietf-teas-ietf-network-slices:
    Framework for IETF Network Slices";
}

identity vpls {
  base service-type;
  description
    "VPLS service.";
  reference
    "RFC 4761: Virtual Private LAN Service (VPLS) Using BGP for
    Auto-Discovery and Signaling
    RFC 4762: Virtual Private LAN Service (VPLS) Using Label
    Distribution Protocol (LDP) Signaling";
}

identity vpws {
  base service-type;
  description
    "Virtual Private Wire Service (VPWS) service.";
  reference
    "RFC 4664: Framework for Layer 2 Virtual Private Networks
    (L2VPNs), Section 3.1.1";
}

identity vpws-evpn {
  base service-type;
  description
    "EVPN used to support VPWS service.";
  reference
    "RFC 8214: Virtual Private Wire Service Support in Ethernet VPN";
}

identity pbb-evpn {
  base service-type;
  description
    "Provider Backbone Bridging (PBB) EVPNs service.";
  reference
    "RFC 7623: Provider Backbone Bridging Combined with Ethernet VPN
    (PBB-EVPN)";
}

identity mpls-evpn {
```

```
    base service-type;
    description
        "MPLS-based EVPN service.";
    reference
        "RFC 7432: BGP MPLS-Based Ethernet VPN";
}

identity vxlan-evpn {
    base service-type;
    description
        "VXLAN-based EVPN service.";
    reference
        "RFC 8365: A Network Virtualization Overlay Solution Using
        Ethernet VPN (EVPN)";
}

identity virtual-network {
    base service-type;
    description
        "Virtual network.";
    reference
        "RFC 8453: Framework for Abstraction and Control of TE
        Networks (ACTN)";
}

/*
Other network service types may be added.
*/

grouping sap-information {
    description
        "Service Attachment Point (SAP) information.";
    list service-attachment-point {
        key "attachment-id";
        description
            "The service attachment points are abstraction of
            the points where network services such as L3VPNs,
            L2VPNs, or network slices can be attached.";
        leaf attachment-id {
            type nt:tp-id;
            description
                "Indicates the name of the interface.";
        }
        leaf interface-type {
            type identityref {
                base if:interface-type;
            }
            config false;
        }
    }
}
```



```
        description
            "The type of the interface.";
    }
    leaf admin-status {
        type boolean;
        description
            "Indicates the administrative status of the SAP.";
    }
    leaf oper-status {
        type boolean;
        description
            "Indicates the operational status.";
    }
    leaf encapsulation-type {
        type identityref {
            base vpn-common:encapsulation-type;
        }
        description
            "Encapsulation type.";
    }
    leaf-list sap-type {
        type identityref {
            base service-type;
        }
        description
            "SAP type.";
    }
    leaf service-description {
        type string;
        description
            "A textual description of the service(s).";
    }
}

augment "/nw:networks/nw:network/nw:network-types" {
    description
        "Introduces a new network type for SAP network.";
    container sap-network {
        presence "Indicates SAP Network Type.";
        description
            "The presence of the container node indicates the
            SAP network type.";
        leaf-list sap-type {
            type identityref {
                base service-type;
            }
            description

```

```
        "Indicates a service type.";
    }
}

augment "/nw:networks/nw:network/nw:node" {
    description
        "Parameters for the service attachment point level.";
    uses sap-information;
}
}
<CODE ENDS>
```

7. IANA Considerations

This document registers the following namespace URI in the "ns" subregistry within the "IETF XML Registry" [RFC3688]:

URI: urn:ietf:params:xml:ns:yang:ietf-sap-ntw
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.

This document registers the following YANG module in the YANG Module Names registry [RFC6020] within the "YANG Parameters" registry:

name: ietf-sap-ntw
namespace: urn:ietf:params:xml:ns:yang:ietf-sap-ntw
maintained by IANA: N
prefix: sap
reference: RFC XXXX

8. Security Considerations

The YANG module specified in this document defines schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

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

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). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

```
* /nw:networks/nw:network/nw:node/sap:service-attachment-point/
   sap:attachment-id
```

This subtree specifies the configurations of the nodes in a SAP network model. Unexpected changes to this subtree could lead to service disruption and/or network misbehavior.

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. These are the subtrees and data nodes and their sensitivity/vulnerability:

```
* /nw:networks/nw:network/nw:node/sap:service-attachment-point
```

Unauthorized access to this subtree can disclose the operational state information of the nodes in a SAP network model.

9. Acknowledgements

Thanks to Adrian Farrell and Daniel King for the suggestions on the names used in a previous version.

10. References

10.1. Normative References

- [I-D.ietf-opsawg-vpn-common]
Barguil, S., Dios, O. G. D., Boucadair, M., and Q. Wu, "A Layer 2/3 VPN Common YANG Model", Work in Progress, Internet-Draft, draft-ietf-opsawg-vpn-common-12, 29 September 2021, <<https://www.ietf.org/archive/id/draft-ietf-opsawg-vpn-common-12.txt>>.
- [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>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [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>>.
- [RFC8345] Clemm, A., Medved, J., Varga, R., Bahadur, N., Ananthakrishnan, H., and X. Liu, "A YANG Data Model for Network Topologies", RFC 8345, DOI 10.17487/RFC8345, March 2018, <<https://www.rfc-editor.org/info/rfc8345>>.
- [RFC8346] Clemm, A., Medved, J., Varga, R., Liu, X., Ananthakrishnan, H., and N. Bahadur, "A YANG Data Model for Layer 3 Topologies", RFC 8346, DOI 10.17487/RFC8346, March 2018, <<https://www.rfc-editor.org/info/rfc8346>>.
- [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>>.

- [RFC8795] Liu, X., Bryskin, I., Beeram, V., Saad, T., Shah, H., and O. Gonzalez de Dios, "YANG Data Model for Traffic Engineering (TE) Topologies", RFC 8795, DOI 10.17487/RFC8795, August 2020, <<https://www.rfc-editor.org/info/rfc8795>>.

10.2. Informative References

- [I-D.ietf-opsawg-l2nm]
Barguil, S., Dios, O. G. D., Boucadair, M., and L. A. Munoz, "A Layer 2 VPN Network YANG Model", Work in Progress, Internet-Draft, draft-ietf-opsawg-l2nm-09, 20 October 2021, <<https://www.ietf.org/archive/id/draft-ietf-opsawg-l2nm-09.txt>>.
- [I-D.ietf-opsawg-l3sm-l3nm]
Barguil, S., Dios, O. G. D., Boucadair, M., Munoz, L. A., and A. Aguado, "A Layer 3 VPN Network YANG Model", Work in Progress, Internet-Draft, draft-ietf-opsawg-l3sm-l3nm-18, 8 October 2021, <<https://www.ietf.org/archive/id/draft-ietf-opsawg-l3sm-l3nm-18.txt>>.
- [I-D.ietf-teas-enhanced-vpn]
Dong, J., Bryant, S., Li, Z., Miyasaka, T., and Y. Lee, "A Framework for Enhanced Virtual Private Network (VPN+) Services", Work in Progress, Internet-Draft, draft-ietf-teas-enhanced-vpn-08, 12 July 2021, <<https://www.ietf.org/archive/id/draft-ietf-teas-enhanced-vpn-08.txt>>.
- [I-D.ietf-teas-ietf-network-slices]
Farrel, A., Gray, E., Drake, J., Rokui, R., Homma, S., Makhijani, K., Contreras, L. M., and J. Tantsura, "Framework for IETF Network Slices", Work in Progress, Internet-Draft, draft-ietf-teas-ietf-network-slices-04, 23 August 2021, <<https://www.ietf.org/archive/id/draft-ietf-teas-ietf-network-slices-04.txt>>.
- [RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <<https://www.rfc-editor.org/info/rfc4364>>.
- [RFC4761] Kompella, K., Ed. and Y. Rekhter, Ed., "Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling", RFC 4761, DOI 10.17487/RFC4761, January 2007, <<https://www.rfc-editor.org/info/rfc4761>>.

- [RFC4762] Lasserre, M., Ed. and V. Kompella, Ed., "Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling", RFC 4762, DOI 10.17487/RFC4762, January 2007, <<https://www.rfc-editor.org/info/rfc4762>>.
- [RFC7149] Boucadair, M. and C. Jacquenet, "Software-Defined Networking: A Perspective from within a Service Provider Environment", RFC 7149, DOI 10.17487/RFC7149, March 2014, <<https://www.rfc-editor.org/info/rfc7149>>.
- [RFC7426] Haleplidis, E., Ed., Pentikousis, K., Ed., Denazis, S., Hadi Salim, J., Meyer, D., and O. Koufopavlou, "Software-Defined Networking (SDN): Layers and Architecture Terminology", RFC 7426, DOI 10.17487/RFC7426, January 2015, <<https://www.rfc-editor.org/info/rfc7426>>.
- [RFC7432] Sajassi, A., Ed., Aggarwal, R., Bitar, N., Isaac, A., Uttaro, J., Drake, J., and W. Henderickx, "BGP MPLS-Based Ethernet VPN", RFC 7432, DOI 10.17487/RFC7432, February 2015, <<https://www.rfc-editor.org/info/rfc7432>>.
- [RFC7623] Sajassi, A., Ed., Salam, S., Bitar, N., Isaac, A., and W. Henderickx, "Provider Backbone Bridging Combined with Ethernet VPN (PBB-EVPN)", RFC 7623, DOI 10.17487/RFC7623, September 2015, <<https://www.rfc-editor.org/info/rfc7623>>.
- [RFC8214] Boutros, S., Sajassi, A., Salam, S., Drake, J., and J. Rabadan, "Virtual Private Wire Service Support in Ethernet VPN", RFC 8214, DOI 10.17487/RFC8214, August 2017, <<https://www.rfc-editor.org/info/rfc8214>>.
- [RFC8299] Wu, Q., Ed., Litkowski, S., Tomotaki, L., and K. Ogaki, "YANG Data Model for L3VPN Service Delivery", RFC 8299, DOI 10.17487/RFC8299, January 2018, <<https://www.rfc-editor.org/info/rfc8299>>.
- [RFC8309] Wu, Q., Liu, W., and A. Farrel, "Service Models Explained", RFC 8309, DOI 10.17487/RFC8309, January 2018, <<https://www.rfc-editor.org/info/rfc8309>>.
- [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>>.
- [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>>.

- [RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <<https://www.rfc-editor.org/info/rfc8343>>.
- [RFC8365] Sajassi, A., Ed., Drake, J., Ed., Bitar, N., Shekhar, R., Uttaro, J., and W. Henderickx, "A Network Virtualization Overlay Solution Using Ethernet VPN (EVPN)", RFC 8365, DOI 10.17487/RFC8365, March 2018, <<https://www.rfc-editor.org/info/rfc8365>>.
- [RFC8453] Ceccarelli, D., Ed. and Y. Lee, Ed., "Framework for Abstraction and Control of TE Networks (ACTN)", RFC 8453, DOI 10.17487/RFC8453, August 2018, <<https://www.rfc-editor.org/info/rfc8453>>.
- [RFC8466] Wen, B., Fioccola, G., Ed., Xie, C., and L. Jalil, "A YANG Data Model for Layer 2 Virtual Private Network (L2VPN) Service Delivery", RFC 8466, DOI 10.17487/RFC8466, October 2018, <<https://www.rfc-editor.org/info/rfc8466>>.
- [RFC8969] Wu, Q., Ed., Boucadair, M., Ed., Lopez, D., Xie, C., and L. Geng, "A Framework for Automating Service and Network Management with YANG", RFC 8969, DOI 10.17487/RFC8969, January 2021, <<https://www.rfc-editor.org/info/rfc8969>>.

Authors' Addresses

Oscar Gonzalez de Dios
Telefonica
Madrid
Spain

Email: oscar.gonzalezdedios@telefonica.com

Samier Barguil
Telefonica
Madrid
Spain

Email: samier.barguilgiraldo.ext@telefonica.com

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing
Jiangsu, 210012
China

Email: bill.wu@huawei.com

Mohamed Boucadair
Orange
France

Email: mohamed.boucadair@orange.com

Victor Lopez
Nokia
Spain

Email: victor.lopez@nokia.com

IPPM Working Group
Internet-Draft
Intended status: Informational
Expires: 28 April 2022

L. Han
M. Wang
China Mobile
F. Yang
J. Huang
Huawei Technologies
25 October 2021

Problem Statement and Requirement for Inband Flow Learning
draft-hwyh-ippm-ps-inband-flow-learning-01

Abstract

Alternate-Marking (coloring) provides a method to perform packet loss, delay, and jitter measurements on live traffic. At the same time, on-path telemetry techniques are used to enable the collection and correlation of performance information to further support autonomous network operations. However, network operators still face the challenge of inband flow identification in large scale deployment. This document addresses the problems by introducing the real network scenarios, and proposes the requirements of supporting inband flow learning of flow information telemetry.

Requirements Language

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.

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 28 April 2022.

Copyright Notice

Copyright (c) 2021 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 Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology	3
3. Problem Statement	3
3.1. Frequent and Dynamic Change of Flows	3
3.1.1. Tidal Effect	4
3.1.2. UPF Expansion	4
3.2. Enterprise Service Demand	4
3.3. Large Scale Network Monitor Deployment and Maintenance	4
3.4. Service Flow Path Change	5
4. Requirement	5
4.1. Ingress Flow Learning	5
4.2. Egress Flow Learning	5
4.3. Hop-by-Hop Flow Learning	6
4.4. Auto Flow Aging	6
4.5. Flow Learning Policy	6
5. IANA Considerations	6
6. Security Considerations	6
7. References	6
7.1. Normative References	6
7.2. Informative References	7
Authors' Addresses	7

1. Introduction

Alternate-Marking (coloring) [RFC8321] provides a method to perform packet loss, delay, and jitter measurements on live traffic. [I-D.ietf-mpls-inband-pm-encapsulation] and [I-D.ietf-6man-ipv6-alt-mark] introduce the MPLS and IPv6 performance measurement applications of alternate marking method respectively, and specifies the encapsulations for MPLS and IPv6. On-path telemetry techniques are used to enable the collection and correlation of performance information from the network. By coloring

the real service flow and telemetry flow statistics, per-flow SLA compliance monitoring becomes available and scalable for network operators. When deployed in network, per-flow monitoring can be applied based on CLI configuration or via Netconf YANG.

However, even though Netconf YANG can provide feasibility to network administration, the characteristic of a flow (e.g. IP 5-tuple) can vary dynamically and mislead the service flow identification. Inband flow learning becomes a challenge in large scale deployment to network operators. This document addresses the problems by introducing the real network scenarios, and proposes the requirements of supporting inband flow learning of flow information telemetry.

2. Terminology

OAM: Operations, Administration, and Maintenance

SLA: Service Level Agreement

NFV: Network Function Virtualization

UNI: User-Network-Interface

CN: Core Network

3. Problem Statement

In an alternate marking application, it is usually to utilize the characteristic fields of packet to identify a service flow. For example, IP 5-tuple is usually to be used as the identifier of a service flow at source node. A concept of flow identifier, such as Flow-ID Label Indicator [I-D.ietf-mpls-inband-pm-encapsulation] or FlowMonID [I-D.ietf-6man-ipv6-alt-mark] is used to identify service flow at transit or egress nodes. The change of packet data fields would mislead the flow identification for flow monitoring and statistics telemetry in large scale.

3.1. Frequent and Dynamic Change of Flows

In 4G/5G mobile backhaul networks, IP address of one service can be changed based on location, time or even with business growth. The following scenarios describes the challenges which 4G/5G mobile service encounters.

3.1.1. Tidal Effect

A Tidal Effect phenomenon has been recognized as traffics between base station and Core Network (CN) show repetitive patterns with spatio-temporal variations. A typical example of Tidal phenomenon is the traffic difference happened in day and night time of a commercial and business area. In day time, eNodeB allocates more core network resources when a large number of user equipment accesses eNodeB, and less resources at night accordingly. The change of the number of UEs and the core network resources may affect the change on source and destination IP address of service flows.

Moreover, NFV used in core network makes the traffic change even worse as the IP address at CN cannot be manually configured or even predicted. In this case, it is impossible for operators to statically deploy flow monitoring and statistics telemetry.

3.1.2. UPF Expansion

In 5G deployment, the increase of number of subscribers triggers the expansion of UPF resources on data plane of 5G core network. After new UPF resource is added, eNodeB sets up a connection to the new UPF. Correspondingly, a new IP flow is created in mobile bearer network. In this scenario, if flow monitoring and statistics telemetry is deployed in a static mode, operators would need to manually add related configurations to mobile bearer network after the core network capacity is expanded, which is very difficult to deploy in practice.

3.2. Enterprise Service Demand

The enterprise services usually connect different private networks between Headquarter and Branches, Branches and Branches. Network operator has very limited or even no information about end users. Besides, information from one site could be changed from time to time. Unpredictable information on enterprise customer side makes impossible for network operators to set up real time flow monitoring, and to avoid the omission of flow monitoring.

3.3. Large Scale Network Monitor Deployment and Maintenance

In a large-scale mobile bearer network, a large number of base stations and corresponding access points may lead to a large number of IP addresses in core network. From network maintenance perspective, when flow monitoring and statistics telemetry is deployed in a static mode, network operator had to manually set up each monitoring instance between base station and core network, then separately delegate configurations to a large number of network

entities. It is difficult for network operators to find an effective way of monitoring creation and maintenance.

Note that traffic monitoring is comprised of uplink and downlink directions, which makes twice of workload on configurations.

3.4. Service Flow Path Change

When a hop-by-hop flow monitoring is required by critical traffic for deep SLA investigation, the actual forwarding path of service flow and the every forwarding nodes along the path are obtained. Network operator delegates different configurations to each node including ingress, transit, and egress nodes on the path.

Once the traffic forwarding path is changed because of service flow switching or route convergence, the monitoring instance on each node needs to be re-deployed on the new path. In this situation, a flexible and efficient deployment approach is required by network operators.

4. Requirement

To face the flow deployment challenges mentioned in preceding section, an approach of inband flow learning is required. It should simplify the deployment of flow monitoring and achieve an automatic mode of telemetry in large scale networks.

4.1. Ingress Flow Learning

On the UNI side of network node, ingress flow learning can help to capture the characteristic data fields of packet and create the monitoring instance when the flow is created from base station. Flexible policy based on access control list (ACL) can facilitate the identification of flow characteristic. For example, IP 2-tuple (DIP+SIP), DSCP value, etc.

4.2. Egress Flow Learning

Similar to the requirement on ingress node, traffic egress node should support the same capability of inband flow learning to create traffic monitoring instance for completing a monitor. When the egress node or egress port of a service flow is changed, the egress node or egress port of service flow can be triggered to re-learn and re-monitor the service flow.

4.3. Hop-by-Hop Flow Learning

When hop-by-hop flow monitoring and telemetry is required, the flow learning and monitor deployment should be created on all the ingress, transit, and egress nodes that service flows pass through. When the path of a service flow changes due to the service switching or network convergence, the service flow re-triggers the flow learning on the new path and starts the new monitoring of service flow.

4.4. Auto Flow Aging

In all the inband flow learning scenarios described above, when the path of a service flow changes, the flow learning on new path is triggered and new monitoring instances are created on devices. Regarding the monitoring instances that have been created before the path change, if there is no traffic detected within a certain period of time, automatic aging and resource recycle should be supported.

4.5. Flow Learning Policy

It is valuable to specify the flow learning policy on equipment when thousands or millions of flows are transmitted. Flow learning policy specifies the metrics and explicit rules executed on equipment, for example the flow is filtered based on a particular range of protocol number. Centralized controller specifies the flow learning policy via management and control plane to equipment, then data plane executes the policies to generate monitoring instance.

5. IANA Considerations

This document has no request to IANA

6. Security Considerations

TBD

7. References

7.1. 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>>.

7.2. Informative References

[I-D.ietf-6man-ipv6-alt-mark]
Fioccola, G., Zhou, T., Cociglio, M., Qin, F., and R. Pang, "IPv6 Application of the Alternate Marking Method", Work in Progress, Internet-Draft, draft-ietf-6man-ipv6-alt-mark-12, 22 October 2021, <<https://www.ietf.org/archive/id/draft-ietf-6man-ipv6-alt-mark-12.txt>>.

[I-D.ietf-mpls-inband-pm-encapsulation]
Cheng, W., Min, X., Zhou, T., Dong, X., and Y. Peleg, "Encapsulation For MPLS Performance Measurement with Alternate Marking Method", Work in Progress, Internet-Draft, draft-ietf-mpls-inband-pm-encapsulation-02, 25 October 2021, <<https://www.ietf.org/archive/id/draft-ietf-mpls-inband-pm-encapsulation-02.txt>>.

[RFC8321] Fioccola, G., Ed., Capello, A., Cociglio, M., Castaldelli, L., Chen, M., Zheng, L., Mirsky, G., and T. Mizrahi, "Alternate-Marking Method for Passive and Hybrid Performance Monitoring", RFC 8321, DOI 10.17487/RFC8321, January 2018, <<https://www.rfc-editor.org/info/rfc8321>>.

Authors' Addresses

Liuyan Han
China Mobile
Beijing
China

Email: hanliuyan@chinamobile.com

Minxue Wang
China Mobile
Beijing
China

Email: wangminxue@chinamobile.com

Fan Yang
Huawei Technologies
Beijing
China

Email: shirley.yangfan@huawei.com

Jinming Huang
Huawei Technologies
Dongguan
China

Email: zhangshengli4@huawei.com

OPSAWG Working Group
Internet-Draft
Intended status: Best Current Practice
Expires: 28 September 2022

M. Richardson
Sandelman Software Works
W. Pan
Huawei Technologies
27 March 2022

Operational Considerations for use of DNS in IoT devices
draft-ietf-opsawg-mud-iot-dns-considerations-04

Abstract

This document details concerns about how Internet of Things devices use IP addresses and DNS names. The issue becomes acute as network operators begin deploying RFC8520 Manufacturer Usage Description (MUD) definitions to control device access.

This document makes recommendations on when and how to use DNS names in MUD files.

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at <https://datatracker.ietf.org/doc/draft-ietf-opsawg-mud-iot-dns-considerations/>.

Discussion of this document takes place on the opsawg Working Group mailing list (<mailto:opsawg@ietf.org>), which is archived at <https://mailarchive.ietf.org/arch/browse/opsawg/>.

Source for this draft and an issue tracker can be found at <https://github.com/mcr/iot-mud-dns-considerations>.

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 28 September 2022.

Copyright Notice

Copyright (c) 2022 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	3
2. Terminology	4
3. Strategies to map names	4
4. DNS and IP Anti-Patterns for IoT device Manufacturers	6
4.1. Use of IP address literals in-protocol	7
4.2. Use of non-deterministic DNS names in-protocol	8
4.3. Use of a too inclusive DNS name	8
5. DNS privacy and outsourcing versus MUD controllers	9
6. Recommendations to IoT device manufacturer on MUD and DNS usage	9
6.1. Consistently use DNS	10
6.2. Use primary DNS names controlled by the manufacturer	10
6.3. Use Content-Distribution Network with stable names	10
6.4. Do not use geofenced names	10
6.5. Prefer DNS servers learnt from DHCP/Route Advertisements	10
7. Privacy Considerations	11
8. Security Considerations	12
9. References	12
9.1. Normative References	13
9.2. Informative References	14
Appendix A. Appendices	15
Authors' Addresses	15

1. Introduction

[RFC8520] provides a standardized way to describe how a specific purpose device makes use of Internet resources. Access Control Lists (ACLs) can be defined in an RFC8520 Manufacturer Usage Description (MUD) file that permit a device to access Internet resources by DNS name.

Use of a DNS name rather than IP address in the ACL has many advantages: not only does the layer of indirection permit the mapping of name to IP address to be changed over time, it also generalizes automatically to IPv4 and IPv6 addresses, as well as permitting load balancing of traffic by many different common ways, including multi-CDN deployments wherein load balancing can account for geography and load.

At the MUD policy enforcement point -- the firewall -- there is a problem. The firewall has only access to the layer-3 headers of the packet. This includes the source and destination IP address, and if not encrypted by IPsec, the destination UDP or TCP port number present in the transport header. The DNS name is not present!

It has been suggested that one answer to this problem is to provide a forced intermediate for the TLS connections. This could in theory be done for TLS 1.2 connections. The MUD policy enforcement point could observe the Server Name Identifier (SNI) [RFC6066]. Some Enterprises do this already. But, as this involves active termination of the TCP connection (a forced circuit proxy) in order to see enough of the traffic, it requires significant effort. But, TLS 1.3 provides options to encrypt the SNI as the ESNI, which renders the practice useless in the end.

So in order to implement these name based ACLs, there must be a mapping between the names in the ACLs and layer-3 IP addresses. The first section of this document details a few strategies that are used.

The second section of this document details how common manufacturer anti-patterns get in the way this mapping.

The third section of this document details how current trends in DNS resolution such as public DNS servers, DNS over TLS (DoT), and DNS over HTTPS (DoH) cause problems for the strategies employed. Poor interactions with content-distribution networks is a frequent pathology that can result.

The fourth section of this document makes a series of recommendations ("best current practices") for manufacturers on how to use DNS, and IP addresses with specific purpose IoT devices.

The Privacy Considerations section concerns itself with issues that DNS-over-TLS and DNS-over-HTTPS are frequently used to deal with. How these concerns apply to IoT devices located within a residence or enterprise is a key concern.

The Security Considerations section covers some of the negative outcomes should MUD/firewall managers and IoT manufacturers choose not to cooperate.

2. Terminology

Although this document is not an IETF Standards Track publication, it adopts the conventions for normative language to provide clarity of instructions to the implementer. 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.

3. Strategies to map names

The most naive method is to try to map IP addresses to names using the in-addr.arpa (IPv4), and ipv6.arpa (IPv6) mappings. This fails for a number of reasons:

1. it can not be done fast enough,
2. it reveals usage patterns of the devices,
3. the mapping are often incomplete,
4. even if the mapping is present, due to virtual hosting, it may not map back to the name used in the ACL.

This is not a successful strategy, its use is NOT RECOMMENDED.

XXX --- explain in detail how this can fail.

XXX --- explain N:1 vs 1:1 for virtual hosting.

The simplest successful strategy for translating names is for a MUD controller to take is to do a DNS lookup on the name (a forward lookup), and then use the resulting IP addresses to populate the physical ACLs.

There are still a number of failures possible.

The most important one is in the mapping of the names to IP addresses may be non-deterministic. [RFC1794] describes the very common mechanism that returns DNS A (or reasonably AAAA) records in a permuted order. This is known as Round Robin DNS, and it has been used for many decades. The device is intended to use the first IP address that is returned, and each query returns addresses in a different ordering, splitting the load among many servers.

This situation does not result in failures as long as all possible A/AAAA records are returned. The MUD controller and the device get a matching set, and the ACLs that are setup cover all possibilities.

There are a number of circumstances in which the list is not exhaustive. The simplest is when the round robin does not return all addresses. This is routinely done by geographical DNS load balancing system. It can also happen if there are more addresses than will conveniently fit into a DNS reply. The reply will be marked as truncated. (If DNSSEC resolution will be done, then the entire RR must be retrieved over TCP (or using a larger EDNS(0) size) before being validated)

However, in a geographical DNS load balancing system, different answers are given based upon the locality of the system asking. There may also be further layers of round-robin indirection.

Aside from the list of records being incomplete, the list may have changed between the time that the MUD controller did the lookup and the time that the IoT device does the lookup, and this change can result in a failure for the ACL to match.

In order to compensate for this, the MUD controller SHOULD regularly do DNS lookups in order to get never have stale data. These lookups need to be rate limited in order to avoid load. It may be necessary to avoid local recursive DNS servers. The MUD controller SHOULD incorporate its own recursive caching DNS server. Properly designed recursive servers should cache data for many minutes to days, while the underlying DNS data can change at a higher frequency, providing different answers to different queries!

A MUD controller that is aware of which recursive DNS server that the IoT device will use can instead query that server on a periodic basis. Doing so provides three advantages:

1. any geographic load balancing will base the decision on the geolocation of the recursive DNS server, and the recursive name server will provide the same answer to the MUD controller as to the IoT device.
2. the resulting name to IP address mapping in the recursive name server will be cached, and will remain the same for the entire advertised Time-To-Live reported in the DNS query return. This also allows the MUD controller to avoid doing unnecessary queries.
3. if any addresses have been omitted in a round-robin DNS process, the cache will have the set of addresses that were returned.

The solution of using the same caching recursive resolver as the target device is very simple when the MUD controllers is located in a residential CPE device. The device is usually also the policy enforcement point for the ACLs, and a caching resolver is typically located on the same device. In addition the convenience, there is a shared fate advantage: as all three components are running on the same device, if the device is rebooted, clearing the cache, then all three components will get restarted when the device is restarted.

Where the solution is more complex is when the MUD controller is located elsewhere in an Enterprise, or remotely in a cloud such as when a Software Defines Network (SDN) is used to manage the ACLs. The DNS servers for a particular device may not be known to the MUD controller, nor the MUD controller be even permitted to make recursive queries that server if it is known. In this case, additional installation specific mechanisms are probably needed to get the right view of DNS.

4. DNS and IP Anti-Patterns for IoT device Manufacturers

In many design fields, there are good patterns that should be emulated, and often there are patterns that should not be emulated. The latter are called anti-patterns, as per [antipatterns].

This section describes a number of things with IoT manufacturers have been observed to do in the field, each of which presents difficulties for MUD enforcement points.

4.1. Use of IP address literals in-protocol

A common pattern for a number of devices is to look for firmware updates in a two step process. An initial query is made (often over HTTPS, sometimes with a POST, but the method is immaterial) to a vendor system that knows whether or not an update is required.

The current firmware model of the device is sometimes provided and then the authoritative server provides a determination if a new version is required, and if so, what version. In simpler cases, an HTTPS end point is queried which provides the name and URL of the most recent firmware.

The authoritative upgrade server then responds with a URL of a firmware blob that the device should download and install. Best practice is that firmware is either signed internally ([I-D.ietf-suit-architecture]) so that it can be verified, or a hash of the blob is provided.

An authoritative server might be tempted to provide an IP address literal inside the protocol: there are two arguments (anti-patterns) for doing this.

One is that it eliminates problems to firmware updates that might be caused by lack of DNS, or incompatibilities with DNS. For instance the bug that causes interoperability issues with some recursive servers would become unpatchable for devices that were forced to use that recursive resolver type.

A second reason to avoid a DNS in the URL is when an inhouse content-distribution system is involved that involves on-demand instances being added (or removed) from a cloud computing architecture.

But, there are many problems with use of IP address literals for the location of the firmware.

The first is that the update service server must decide whether to provide an IPv4 or an IPv6 literal. A DNS name can contain both kinds of addresses, and can also contain many different IP addresses of each kind.

The second problem is that it forces the MUD file definition to contain the exact same IP address literals. It must also contain an ACL for each address literal. DNS provides a useful indirection method that naturally aggregates the addresses.

A third problem involves the use of HTTPS. IP address literals do not provide enough context for TLS ServerNameIndicator to be useful [RFC6066]. This limits the firmware repository to be a single tenant on that IP address, and for IPv4 (at least), this is no longer a sustainable use of IP addresses.

And with any non-deterministic name or address that is returned, the MUD controller is not challenged to validate the transaction, as it can not see into the communication.

Third-party content-distribution networks (CDN) tend to use DNS names in order to isolate the content-owner from changes to the distribution network.

4.2. Use of non-deterministic DNS names in-protocol

A second pattern is for a control protocol to connect to a known HTTP end point. This is easily described in MUD. Within that control protocol references are made to additional content at other URLs. The values of those URLs do not fit any easily described pattern and may point at arbitrary names.

Those names are often within some third-party Content-Distribution-Network (CDN) system, or may be arbitrary names in a cloud-provider storage system such as Amazon S3 (such [AmazonS3], or [Akamai]).

Such names may be unpredictably chosen by the content provider, and not the content owner, and so impossible to insert into a MUD file.

Even if the content provider chosen names are deterministic they may change at a rate much faster than MUD files can be updated.

This in particular may apply to the location where firmware updates may be retrieved.

4.3. Use of a too inclusive DNS name

Some CDNs make all customer content at a single URL (such as s3.amazonaws.com). This seems to be ideal from a MUD point of view: a completely predictable URL.

The problem is that a compromised device could then connect to the contents of any bucket, potentially attacking the data from other customers.

Exactly what the risk is depends upon what the other customers are doing: it could be limited to simply causing a distributed denial of service attack resulting to high costs to those customers, or such an attack could potentially include writing content.

Amazon has recognized the problems associated with this practice, and aims to change it to a virtual hosting model, as per [awss3virtualhosting].

The MUD ACLs provide only for permitting end points (hostnames and ports), but do not filter URLs (nor could filtering be enforced within HTTPS).

5. DNS privacy and outsourcing versus MUD controllers

[RFC7858] and [RFC8094] provide for DNS over TLS (DoT) and DNS over HTTPS (DoH). [I-D.ietf-dnsop-terminology-ter] details the terms. But, even with traditional DNS over Port-53 (Do53), it is possible to outsource DNS queries to other public services, such as those operated by Google, CloudFlare, Verisign, etc.

For some users and classes of device, revealing the DNS queries to those outside entities may constitute a privacy concern. For other users the use of an insecure local resolver may constitute a privacy concern.

As described above in Section 3 the MUD controller needs to have access to the same resolver(s) as the IoT device.

6. Recommendations to IoT device manufacturer on MUD and DNS usage

Inclusion of a MUD file with IoT devices is operationally quite simple. It requires only a few small changes to the DHCP client code to express the MUD URL. It can even be done without code changes via the use of a QR code affixed to the packaging (see [I-D.richardson-mud-qrcode]).

The difficult part is determining what to put into the MUD file itself. There are currently tools that help with the definition and analysis of MUD files, see [mudmaker]. The remaining difficulty is now the semantic contents of what is in the MUD file. An IoT manufacturer must now spend some time reviewing what the network communications that their device does.

This document has discussed a number of challenges that occur relating to how DNS requests are made and resolved, and it is the goal of this section to make recommendations on how to modify IoT systems to work well with MUD.

6.1. Consistently use DNS

For the reasons explained in Section 4.1, the most important recommendation is to avoid using IP address literals in any protocol. Names should always be used.

6.2. Use primary DNS names controlled by the manufacturer

The second recommendation is to allocate and use names within zones controlled by the manufacturer. These names can be populated with an alias (see [RFC8499] section 2) that points to the production system. Ideally, a different name is used for each logical function, allowing for different rules in the MUD file to be enabled and disabled.

While it used to be costly to have a large number of aliases in a web server certificate, this is no longer the case. Wildcard certificates are also commonly available which allowed for an infinite number of possible names.

6.3. Use Content-Distribution Network with stable names

When aliases point to a Content-Distribution Network (CDN), prefer to use stable names that point to appropriately load balanced targets. CDNs that employ very low time-to-live (TTL) values for DNS make it harder for the MUD controller to get the same answer as the IoT Device. A CDN that always returns the same set of A and AAAA records, but permutes them to provide the best one first provides a more reliable answer.

6.4. Do not use geofenced names

Due the problems with different answers from different DNS servers, described above, a strong recommendation is to avoid using such things.

6.5. Prefer DNS servers learnt from DHCP/Route Advertisements

IoT Devices should prefer doing DNS to the network provided DNS servers. Whether this is restricted to Classic DNS (Do53) or also includes using DoT/DoH is a local decision, but a locally provided DoT server SHOULD be used, as recommended by [I-D.reddy-dprive-bootstrap-dns-server] and [I-D.peterson-doh-dhcp].

The ADD WG is currently only focusing on insecure discovery mechanisms like DHCP/RA [I-D.btw-add-home] and DNS based discovery mechanisms ({I-D.pauly-add-deer}). Secure discovery of network provided DoH/DoT resolver is possible using the mechanisms discussed in [I-D.reddy-add-enterprise] section-4.

Use of public QuadX resolver instead of the provided DNS resolver, whether Do53, DoT or DoH is discouraged. Should the network provide such a resolver for use, then there is no reason not to use it, as the network operator has clearly thought about this.

Some manufacturers would like to have a fallback to using a public resolver to mitigate against local misconfiguration. There are a number of reasons to avoid this, or at least do this very carefully.

It is recommended that use of non-local resolvers is only done when the locally provided resolvers provide no answers to any queries at all, and do so repeatedly. The use of the operator provided resolvers SHOULD be retried on a periodic basis, and once they answer, there should be no further attempts to contact public resolvers.

Finally, the list of public resolvers that might be contacted MUST be listed in the MUD file as destinations that are to be permitted! This should include the port numbers (53, 853 for DoT, 443 for DoH) that will be used as well.

7. Privacy Considerations

The use of non-local DNS servers exposes the list of names resolved to a third parties, including passive eavesdroppers.

The use of DoT and DoH eliminates the minimizes threat from passive eavesdropped, but still exposes the list to the operator of the DoT or DoH server. There are additional methods, such as described by [I-D.pauly-dprive-oblivious-doh].

The use of unencrypted (Do53) requests to a local DNS server exposes the list to any internal passive eavesdroppers, and for some situations that may be significant, particularly if unencrypted WiFi is used. Use of Encrypted DNS connection to a local DNS recursive resolver is a preferred choice, assuming that the trust anchor for the local DNS server can be obtained, such as via [I-D.reddy-dprive-bootstrap-dns-server].

IoT devices that reach out to the manufacturer at regular intervals to check for firmware updates are informing passive eavesdroppers of the existence of a specific manufacturer's device being present at the origin location.

Identifying the IoT device type empowers the attacker to launch targeted attacks to the IoT device (e.g., Attacker can advantage of the device vulnerability).

While possession of a Large (Kitchen) Appliance at a residence may be uninteresting to most, possession of intimate personal devices (e.g., "sex toys") may be a cause for embarrassment.

IoT device manufacturers are encouraged to find ways to anonymize their update queries. For instance, contracting out the update notification service to a third party that deals with a large variety of devices would provide a level of defense against passive eavesdropping. Other update mechanisms should be investigated, including use of DNSSEC signed TXT records with current version information. This would permit DoT or DoH to convey the update notification in a private fashion. This is particularly powerful if a local recursive DoT server is used, which then communicates using DoT over the Internet.

The more complex case of section Section 4.1 postulates that the version number needs to be provided to an intelligent agent that can decide the correct route to do upgrades. The current [I-D.ietf-suit-architecture] specification provides a wide variety of ways to accomplish the same thing without having to divulge the current version number.

The use of a publically specified firmware update protocol would also enhance privacy of IoT devices. In such a system the IoT device would never contact the manufacturer for version information or for firmware itself. Instead, details of how to query and where to get the firmware would be provided as a MUD extension, and a Enterprise-wide mechanism would retrieve firmware, and then distribute it internally. Aside from the bandwidth savings of downloading the firmware only once, this also makes the number of devices active confidential, and provides some evidence about which devices have been upgraded and which ones might still be vulnerable. (The unpatched devices might be lurking, powered off, lost in a closet)

8. Security Considerations

This document deals with conflicting Security requirements:

1. devices which an operator wants to manage using [RFC8520]
2. requirements for the devices to get access to network resources that may be critical to their continued safe operation.

This document takes the view that the two requirements do not need to be in conflict, but resolving the conflict requires some advance planning by all parties.

9. References

9.1. Normative References

- [Akamai] "Akamai", 2019,
<https://en.wikipedia.org/wiki/Akamai_Technologies>.
- [AmazonS3] "Amazon S3", 2019,
<https://en.wikipedia.org/wiki/Amazon_S3>.
- [I-D.ietf-dnsop-terminology-ter]
Hoffman, P., "Terminology for DNS Transports and Location", Work in Progress, Internet-Draft, draft-ietf-dnsop-terminology-ter-02, 3 August 2020,
<<https://www.ietf.org/archive/id/draft-ietf-dnsop-terminology-ter-02.txt>>.
- [I-D.ietf-suit-architecture]
Moran, B., Tschofenig, H., Brown, D., and M. Meriac, "A Firmware Update Architecture for Internet of Things", Work in Progress, Internet-Draft, draft-ietf-suit-architecture-16, 27 January 2021, <<https://www.ietf.org/archive/id/draft-ietf-suit-architecture-16.txt>>.
- [I-D.peterson-doh-dhcp]
Peterson, T., "DNS over HTTP resolver announcement Using DHCP or Router Advertisements", Work in Progress, Internet-Draft, draft-peterson-doh-dhcp-01, 21 October 2019, <<https://www.ietf.org/archive/id/draft-peterson-doh-dhcp-01.txt>>.
- [I-D.reddy-dprive-bootstrap-dns-server]
Reddy, T., Wing, D., Richardson, M. C., and M. Boucadair, "A Bootstrapping Procedure to Discover and Authenticate DNS-over-TLS and DNS-over-HTTPS Servers", Work in Progress, Internet-Draft, draft-reddy-dprive-bootstrap-dns-server-08, 6 March 2020,
<<https://www.ietf.org/archive/id/draft-reddy-dprive-bootstrap-dns-server-08.txt>>.
- [RFC1794] Brisco, T., "DNS Support for Load Balancing", RFC 1794, DOI 10.17487/RFC1794, April 1995,
<<https://www.rfc-editor.org/info/rfc1794>>.
- [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>>.

- [RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, DOI 10.17487/RFC6146, April 2011, <<https://www.rfc-editor.org/info/rfc6146>>.
- [RFC7858] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)", RFC 7858, DOI 10.17487/RFC7858, May 2016, <<https://www.rfc-editor.org/info/rfc7858>>.
- [RFC8094] Reddy, T., Wing, D., and P. Patil, "DNS over Datagram Transport Layer Security (DTLS)", RFC 8094, DOI 10.17487/RFC8094, February 2017, <<https://www.rfc-editor.org/info/rfc8094>>.
- [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>>.
- [RFC8499] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", BCP 219, RFC 8499, DOI 10.17487/RFC8499, January 2019, <<https://www.rfc-editor.org/info/rfc8499>>.
- [RFC8520] Lear, E., Droms, R., and D. Romascanu, "Manufacturer Usage Description Specification", RFC 8520, DOI 10.17487/RFC8520, March 2019, <<https://www.rfc-editor.org/info/rfc8520>>.

9.2. Informative References

- [antipatterns] "AntiPattern", 12 July 2021, <<https://www.agilealliance.org/glossary/antipattern>>.
- [awss3virtualhosting] "Down to the Wire: AWS Delays 'Path-Style' S3 Deprecation at Last Minute", 12 July 2021, <<https://techmonitor.ai/techonology/cloud/aws-s3-path-deprecation>>.
- [I-D.btw-add-home] Boucadair, M., Reddy, T., Wing, D., Cook, N., and T. Jensen, "DHCP and Router Advertisement Options for Encrypted DNS Discovery", Work in Progress, Internet-Draft, draft-btw-add-home-12, 22 January 2021, <<https://www.ietf.org/archive/id/draft-btw-add-home-12.txt>>.

[I-D.pauly-dprive-oblivious-doh]

Kinnear, E., McManus, P., Pauly, T., Verma, T., and C. A. Wood, "Oblivious DNS Over HTTPS", Work in Progress, Internet-Draft, draft-pauly-dprive-oblivious-doh-11, 17 February 2022, <<https://www.ietf.org/archive/id/draft-pauly-dprive-oblivious-doh-11.txt>>.

[I-D.reddy-add-enterprise]

Reddy, T. and D. Wing, "DNS-over-HTTPS and DNS-over-TLS Server Deployment Considerations for Enterprise Networks", Work in Progress, Internet-Draft, draft-reddy-add-enterprise-00, 23 June 2020, <<https://www.ietf.org/archive/id/draft-reddy-add-enterprise-00.txt>>.

[I-D.richardson-mud-qrcode]

Richardson, M., Latour, J., and H. H. Gharakheili, "On loading MUD URLs from QR codes", Work in Progress, Internet-Draft, draft-richardson-mud-qrcode-07, 21 March 2022, <<https://www.ietf.org/archive/id/draft-richardson-mud-qrcode-07.txt>>.

[mudmaker] "Mud Maker", 2019, <<https://mudmaker.org>>.

[RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <<https://www.rfc-editor.org/info/rfc6066>>.

Appendix A. Appendices

Authors' Addresses

Michael Richardson
Sandelman Software Works
Email: mcr+ietf@sandelman.ca

Wei Pan
Huawei Technologies
Email: william.panwei@huawei.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: 28 April 2022

G. Harris, Ed.
M. Richardson
Sandelman
25 October 2021

PCAP Capture File Format
draft-ietf-opsawg-pcap-00

Abstract

This document describes the format used by the libpcap library to record captured packets to a file. Programs using the libpcap library to read and write those files, and thus reading and writing files in that format, include tcpdump.

Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document takes place on the OPSAWG Working Group mailing list (opsawg@ietf.org), which is archived at <https://mailarchive.ietf.org/arch/browse/opsawg/>.

Source for this draft and an issue tracker can be found at <https://github.com/pcapng/pcapng>.

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 28 April 2022.

Copyright Notice

Copyright (c) 2021 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 Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology	3
3. General File Structure	3
4. File Header	3
5. Packet Record	5
6. Recommended File Name Extension: .pcap	7
7. Security Considerations	7
8. IANA Considerations	7
8.1. Media-Type Registry	7
8.1.1. application/pcap	7
8.2. LinkType Registry	8
9. Contributors	34
10. Acknowledgments	35
11. References	35
11.1. Normative References	35
11.2. Informative References	35
Authors' Addresses	35

1. Introduction

In the late 1980's, Van Jacobson, Steve McCanne, and others at the Network Research Group at Lawrence Berkeley National Laboratory developed the tcpdump program to capture and dissect network traces. The code to capture traffic, using low-level mechanisms in various operating systems, and to read and write network traces to a file was later put into a library named libpcap.

This document describes the format used by tcpdump, and other programs using libpcap, to read and write network traces.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. General File Structure

A capture file begins with a File Header, followed by zero or more Packet Records, one per packet.

All fields in the File Header and in Packet Records will always be saved according to the characteristics (little endian / big endian) of the capturing machine. This refers to all the fields that are saved as numbers and that span over two or more octets.

The approach of having the file saved in the native format of the generating host is more efficient because it avoids translation of data when reading / writing on the host itself, which is the most common case when generating/processing capture captures.

The packets are shown in traditional IETF diagram, with the bits numbered from the left to the right. The bit numbering does not reflect the binary value position, as IETF protocols are traditionally in big-endian network-byte order. The most significant bit is therefore on the left in this diagram as if the file is being stored on a big-endian system.

4. File Header

The File Header has the following format:

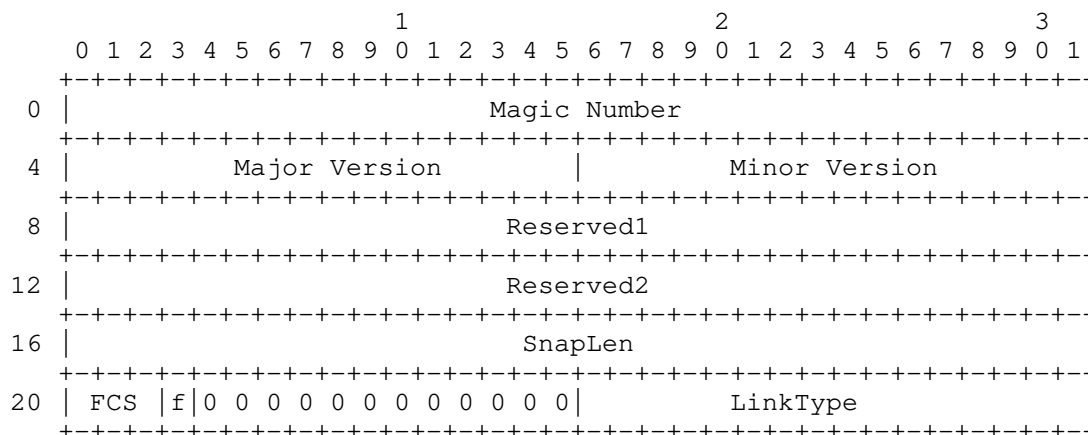


Figure 1: File Header

The File Header length is 24 octets.

The meaning of the fields in the File Header is:

Magic Number (32 bits): an unsigned magic number, whose value is either the hexadecimal number 0xA1B2C3D4 or the hexadecimal number 0xA1B23C4D.

If the value is 0xA1B2C3D4, time stamps in Packet Records (see Figure 2) are in seconds and microseconds; if it is 0xA1B23C4D, time stamps in Packet Records are in seconds and nanoseconds.

These numbers can be used to distinguish sessions that have been saved on little-endian machines from the ones saved on big-endian machines, and to heuristically identify pcap files.

Major Version (16 bits): an unsigned value, giving the number of the current major version of the format. The value for the current version of the format is 2. This value should change if the format changes in such a way that code that reads the new format could not read the old format (i.e., code to read both formats would have to check the version number and use different code paths for the two formats) and code that reads the old format could not read the new format.

Minor Version (16 bits): an unsigned value, giving the number of the

current minor version of the format. The value is for the current version of the format is 4. This value should change if the format changes in such a way that code that reads the new format could read the old format without checking the version number but code that reads the old format could not read all files in the new format.

Reserved1 (32 bits): not used - SHOULD be filled with 0 by pcap file writers, and MUST be ignored by pcap file readers. This value was documented by some older implementations as "gmt to local correction". Some older pcap file writers stored non-zero values in this field.

Reserved2 (32 bits): not used - SHOULD be filled with 0 by pcap file writers, and MUST be ignored by pcap file readers. This value was documented by some older implementations as "accuracy of timestamps". Some older pcap file writers stored non-zero values in this field.

SnapLen (32 bits): an unsigned value indicating the maximum number of octets captured from each packet. The portion of each packet that exceeds this value will not be stored in the file. This value MUST NOT be zero; if no limit was specified, the value should be a number greater than or equal to the largest packet length in the file.

LinkType (16 bits): a 16-bit unsigned value that defines the link layer type of packets in the file. This field is defined in the Section 8.2 IANA registry.

Frame Cyclic Sequence (FCS) present (4 bits): if the "f" bit is set, then the 3 FCS bits provide the number of 16-bit (2 byte) words of FCS that are appended to each packet.

valid values are between 0 and 7, with ethernet typically having a length of 4 bytes, or a value of 2.

The bits marked as zero MUST be set to zero by pcap writers, and MUST be ignored by pcap readers.

5. Packet Record

A Packet Record is the standard container for storing the packets coming from the network.

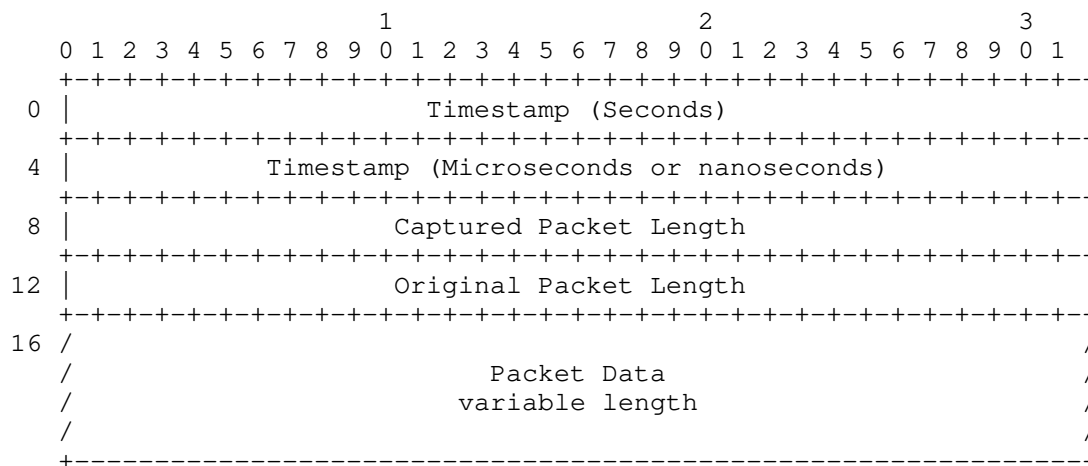


Figure 2: Packet Record

The Packet Header length is 16 octets.

The meaning of the fields in the Packet Record is:

Timestamp (Seconds) and Timestamp (Microseconds or nanoseconds): seconds and fraction of a seconds values of a timestamp.

The seconds value is a 32-bit unsigned integer that represents the number of seconds that have elapsed since 1970-01-01 00:00:00 UTC, and the microseconds or nanoseconds value represents the number of microseconds or nanoseconds that have elapsed since that seconds.

Whether the value represents microseconds or nanoseconds is specified by the magic number in the File Header.

Captured Packet Length (32 bits): an unsigned value that indicates the number of octets captured from the packet (i.e. the length of the Packet Data field). It will be the minimum value among the Original Packet Length and the snapshot length for the interface (SnapLen, defined in Figure 1).

Original Packet Length (32 bits): an unsigned value that indicates the actual length of the packet when it was transmitted on the network. It can be different from the Captured Packet Length if the packet has been truncated by the capture process.

Packet Data: the data coming from the network, including link-layer

headers. The actual length of this field is Captured Packet Length. The format of the link-layer headers depends on the LinkType field specified in the file header (see Figure 1) and it is specified in the entry for that format in [LINKTYPES].

6. Recommended File Name Extension: .pcap

The recommended file name extension for the "PCAP Capture File Format" specified in this document is ".pcap".

On Windows and macOS, files are distinguished by an extension to their filename. Such an extension is technically not actually required, as applications should be able to automatically detect the pcap file format through the "magic bytes" at the beginning of the file, as some other UN*X desktop environments do. However, using name extensions makes it easier to work with files (e.g. visually distinguish file formats) so it is recommended - though not required - to use .pcap as the name extension for files following this specification.

Please note: To avoid confusion (such as the current usage of .cap for a plethora of different capture file formats) file name extensions other than .pcap should be avoided.

There is new work to create the PCAP Next Generation capture File Format (see [I-D.tuexen-opsawg-pcapng]). The new file format is not compatible with this specification, but many programs read both transparently. Files of that type will usually start with a Section Header Block, with a magic number of 0x0A0D0D0A.

7. Security Considerations

TBD.

8. IANA Considerations

This document requires the following IANA actions:

8.1. Media-Type Registry

This section registers the the 'application/pcap' in the "Media Types" registry. These media types are used to indicate that the content is packet capture as described in this document.

8.1.1. application/pcap

Type name: application
Subtype name: pcap
Required parameters: none
Optional parameters: none
Encoding considerations: PCAP files contain network packets
Security considerations: See Security Considerations, Section
Interoperability considerations: The format is designed to be broadly interoperable.
Published specification: THIS RFC.
Applications that use this media type: tcpdump, Wireshark, others.
Additional information:
 Magic number(s): 0xA1B2C3D4, and 0xA1B23C4D in both endian orders
 File extension(s): .pcap
 Macintosh file type code(s): none
Person & email address to contact for further information: The Tcpdump Group, www.tcpdump.org
Intended usage: LIMITED
Restrictions on usage: NONE
Author: Guy Harris and Michael Richardson
Change controller: The Tcpdump Group
Provisional registration? (standards tree only): NO

8.2. LinkType Registry

IANA is requested to create a new Registry entitled: "The PCAP Registry", and within that Registry to create a table called: "PCAP LinkType List".

The LinkType Registry is a table of 16-bit numbers. The Registry has three sections with different [RFC8126] rules:

- * values from 0 to 32767 are marked as Specification Required.
- * values from 32768 to 65000 are marked as First-Come First-Served.
- * values from 65000 to 65535 are marked as Private Use.

The Registry has four columns: the symbolic name (LINKTYPE_something), the integer value, a very short description, and the document/requestor reference.

The Registry shall be populated as follows in the table below. In each case here, the reference should be <http://www.tcpdump.org/linktypes.html>, which is not repeated.

The initial value of table is based upon the Link type list maintained by libpcap, and published on the [tcpdump.org](http://www.tcpdump.org) web site as <http://www.tcpdump.org/linktypes.html>.

There is often an associated DLT value which are often identical in value, but not universally so. DLT values are associated with specific operation system captures, and are operating system specific.

LINKTYPE name	LINKTYPE value	description
LINKTYPE_NULL	0	BSD loopback encapsulation
LINKTYPE_ETHERNET	1	IEEE 802.3 Ethernet
LINKTYPE_EXP_ETHERNET	2	Xerox experimental 3Mb Ethernet
LINKTYPE_AX25	3	AX.25 packet
LINKTYPE_PRONET	4	Reserved for PRONET
LINKTYPE_CHAOS	5	Reserved for MIT CHAOSNET
LINKTYPE_IEEE802_5	6	IEEE 802.5 Token Ring
LINKTYPE_ARCNET_BSD	7	ARCNET Data Packets with BSD encapsulation
LINKTYPE_SLIP	8	SLIP, w/ LINKTYPE_SLIP header.
LINKTYPE_PPP	9	PPP, as per RFC 1661/RFC 1662
LINKTYPE_FDDI	10	FDDI: per ANSI INCITS 239-1994.
not to be used	11-49	Do not use these values
LINKTYPE_PPP_HDLC	50	PPP in HDLC-like framing, as per RFC 1662

LINKTYPE_PPP_ETHER	51	PPPoE; per RFC 2516
not to be used	52-98	Do not use these values
LINKTYPE_SYMANTEC_FIREWALL	99	Reserved for Symantec Enterprise Firewall
LINKTYPE_ATM_RFC1483	100	RFC 1483 LLC/SNAP-encapsulated ATM
LINKTYPE_RAW	101	Raw IP; begins with an IPv4 or IPv6 header
LINKTYPE_SLIP_BSDOS	102	Reserved for BSD/OS SLIP BPF header
LINKTYPE_PPP_BSDOS	103	Reserved for BSD/OS PPP BPF header
LINKTYPE_C_HDLC	104	Cisco PPP with HDLC framing, as per section 4.3.1 of RFC 1547
LINKTYPE_IEEE802_11	105	IEEE 802.11 wireless LAN.
LINKTYPE_ATM_CLIP	106	ATM Classical IP, with no header preceding IP
LINKTYPE_FRELAY	107	Frame Relay LAPF frames
LINKTYPE_LOOP	108	OpenBSD loopback encapsulation
LINKTYPE_ENC	109	Reserved for OpenBSD IPSEC encapsulation
LINKTYPE_LANE8023	110	Reserved for ATM LANE + 802.3
LINKTYPE_HIPPI	111	Reserved for NetBSD HIPPI

LINKTYPE_HDLC	112	Reserved for NetBSD HDLC framing
LINKTYPE_LINUX_SLL	113	Linux "cooked" capture encapsulation
LINKTYPE_LTALK	114	Apple LocalTalk
LINKTYPE_ECONET	115	Reserved for Acorn Econet
LINKTYPE_IPFILTER	116	Reserved for OpenBSD ipfilter
LINKTYPE_PFLOG	117	OpenBSD pflog; "struct pfloghdr" structure
LINKTYPE_CISCO_IOS	118	Reserved for Cisco-internal use
LINKTYPE_IEEE802_11_PRISM	119	Prism monitor mode
LINKTYPE_IEEE802_11_AIRONET	120	Reserved for 802.11 + FreeBSD Aironet radio metadata
LINKTYPE_HHDLC	121	Reserved for Siemens HiPath HDLC
LINKTYPE_IP_OVER_FC	122	RFC 2625 IP-over-Fibre Channel
LINKTYPE_SUNATM	123	ATM traffic, / per SunATM devices
LINKTYPE_RIO	124	Reserved for RapidIO
LINKTYPE_PCI_EXP	125	Reserved for PCI Express
LINKTYPE_AURORA	126	Reserved for Xilinx Aurora link layer
LINKTYPE_IEEE802_11_RADIOTAP	127	Radiotap

		header[Radiotap], followed by an 802.11 header
LINKTYPE_TZSP	128	Reserved for Tazmen Sniffer Protocol
LINKTYPE_ARCNET_LINUX	129	ARCNET Data Packets, per RFC 1051 frames w/variations
LINKTYPE_JUNIPER_MLPPP	130	Reserved for Juniper Networks
LINKTYPE_JUNIPER_MLFR	131	Reserved for Juniper Networks
LINKTYPE_JUNIPER_ES	132	Reserved for Juniper Networks
LINKTYPE_JUNIPER_GGSN	133	Reserved for Juniper Networks
LINKTYPE_JUNIPER_MFR	134	Reserved for Juniper Networks
LINKTYPE_JUNIPER_ATM2	135	Reserved for Juniper Networks
LINKTYPE_JUNIPER_SERVICES	136	Reserved for Juniper Networks
LINKTYPE_JUNIPER_ATM1	137	Reserved for Juniper Networks
LINKTYPE_APPLE_IP_OVER_IEEE1394	138	Apple IP-over-IEEE 1394 cooked header
LINKTYPE_MTP2_WITH_PHDR	139	Signaling System 7 (SS7) Message Transfer Part Level ITU-T Q.703
LINKTYPE_MTP2	140	SS7 Level 2, Q.703
LINKTYPE_MTP3	141	SS7 Level 3, Q.704
LINKTYPE_SCCP	142	SS7 Control Part,

		ITU-T Q.711/Q.712/Q.713/ Q.714
LINKTYPE_DOCSIS	143	DOCSIS MAC frames, DOCSIS 3.1
LINKTYPE_LINUX_IRDA	144	Linux-IrDA packets w/ LINKTYPE_LINUX_IRDA header
LINKTYPE_IBM_SP	145	Reserved for IBM SP switch
LINKTYPE_IBM_SN	146	Reserved for IBM Next Federation switch
LINKTYPE_RESERVED_01	147	For private use
LINKTYPE_RESERVED_02	148	For private use
LINKTYPE_RESERVED_03	149	For private use
LINKTYPE_RESERVED_04	150	For private use
LINKTYPE_RESERVED_05	151	For private use
LINKTYPE_RESERVED_06	152	For private use
LINKTYPE_RESERVED_07	153	For private use
LINKTYPE_RESERVED_08	154	For private use
LINKTYPE_RESERVED_09	155	For private use
LINKTYPE_RESERVED_10	156	For private use
LINKTYPE_RESERVED_11	157	For private use
LINKTYPE_RESERVED_12	158	For private use
LINKTYPE_RESERVED_13	159	For private use
LINKTYPE_RESERVED_14	160	For private use
LINKTYPE_RESERVED_15	161	For private use

LINKTYPE_RESERVED_16	162	For private use
LINKTYPE_IEEE802_11_AVS	163	AVS header[AVS], followed by an 802.11 header
LINKTYPE_JUNIPER_MONITOR	164	Reserved for Juniper Networks
LINKTYPE_BACNET_MS_TP	165	BACnet MS/TP frames, per 9.3 MS/TP Frame Format ANSI 135
LINKTYPE_PPP_PPPD	166	PPP in HDLC-like encapsulation, like LINKTYPE_PPP_HDLC, different stuffing
LINKTYPE_JUNIPER_PPPOE	167	Reserved for Juniper Networks
LINKTYPE_JUNIPER_PPPOE_ATM	168	Reserved for Juniper Networks
LINKTYPE_GPRS_LLC	169	General Packet Radio Service Logical Link Control, as per 3GPP TS 04.64
LINKTYPE_GPF_T	170	Transparent-mapped generic framing procedure, as specified by ITU-T Recommendation G.7041/Y.1303
LINKTYPE_GPF_F	171	Frame-mapped generic framing procedure, as specified by ITU-T Recommendation G.7041/Y.1303
LINKTYPE_GCOM_T1E1	172	Reserved for Gcom T1/E1 line monitoring equipment
LINKTYPE_GCOM_SERIAL	173	Reserved for Gcom

		T1/E1 line monitoring equipment
LINKTYPE_JUNIPER_PIC_PEER	174	Reserved for Juniper Networks
LINKTYPE_ERF_ETH	175	Endace ERF header followed by 802.3 Ethernet
LINKTYPE_ERF_POS	176	Endace ERF header followed by Packet-over-SONET
LINKTYPE_LINUX_LAPD	177	Link Access Procedures on the D Channel (LAPD) frames, as specified by ITU-T Recommendation Q.920 and ITU-T Recommendation Q.921, captured via vISDN, with a LINKTYPE_LINUX_LAPD header, followed by the Q.921 frame, starting with the address field.
LINKTYPE_JUNIPER_ETHER	178	Reserved for Juniper Networks
LINKTYPE_JUNIPER_PPP	179	Reserved for Juniper Networks
LINKTYPE_JUNIPER_FRELAY	180	Reserved for Juniper Networks
LINKTYPE_JUNIPER_CHDLC	181	Reserved for Juniper Networks
LINKTYPE_MFR	182	FRF.16.1 Multi-Link Frame Relay frames, beginning with an FRF.12 Interface fragmentation format fragmentation

		header.
LINKTYPE_JUNIPER_VP	182	Reserved for Juniper Networks
LINKTYPE_A653_ICM	185	Reserved for Arinc 653 Interpartition Communication messages
LINKTYPE_USB_FREEBSD	186	USB packets, beginning with a FreeBSD USB header
LINKTYPE_BLUETOOTH_HCI_H4	187	Bluetooth HCI UART transport layer; the frame contains an HCI packet indicator byte, as specified by the UART Transport Layer portion of the most recent Bluetooth Core specification, followed by an HCI packet of the specified packet type, as specified by the Host Controller Interface Functional Specification portion of the most recent Bluetooth Core Specification.
LINKTYPE_IEEE802_16_MAC_CPS	188	Reserved for IEEE 802.16 MAC Common Part Sublayer
LINKTYPE_USB_LINUX	189	USB packets, beginning with a Linux USB header, as specified by the struct usbmon_packet in the Documentation/usb/usbmon.txt file in

		the Linux source tree. Only the first 48 bytes of that header are present. All fields in the header are in host byte order. When performing a live capture, the host byte order is the byte order of the machine on which the packets are captured. When reading a pcap file, the byte order is the byte order for the file, as specified by the file's magic number; when reading a pcapng file, the byte order is the byte order for the section of the pcapng file, as specified by the Section Header Block.
LINKTYPE_CAN20B	190	Reserved for Controller Area Network (CAN) v. 2.0B packets
LINKTYPE_IEEE802_15_4_LINUX	191	IEEE 802.15.4, with address fields padded, as is done by Linux drivers
LINKTYPE_PPI	192	Per-Packet Information information, as specified by the Per-Packet Information Header Specification , followed by a packet

		with the LINKTYPE_ value specified by the pph_dlt field of that header.
LINKTYPE_IEEE802_16_MAC_CPS_RADIO	193	Reserved for 802.16 MAC Common Part Sublayer plus radio header
LINKTYPE_JUNIPER_ISM	194	Reserved for Juniper Networks
LINKTYPE_IEEE802_15_4_WITHFCS	195	IEEE 802.15.4 Low-Rate Wireless Networks, with each packet having the FCS at the end of the frame.
LINKTYPE_SITA	196	Various link-layer types, with a pseudo-header , for SITA
LINKTYPE_ERF	197	Various link-layer types, with a pseudo-header, for Endace DAG cards; encapsulates Endace ERF records.
LINKTYPE_RAIF1	198	Reserved for Ethernet packets captured from a u10 Networks board
LINKTYPE_IPMB_KONTRON	199	Reserved for IPMB packet for IPMI, with a 2-byte header
LINKTYPE_JUNIPER_ST	200	Reserved for Juniper Networks
LINKTYPE_BLUETOOTH_HCI_H4_WITH_PHDR	201	Bluetooth HCI UART transport layer; the frame contains a 4-byte direction

		field, in network byte order (big-endian), the low-order bit of which is set if the frame was sent from the host to the controller and clear if the frame was received by the host from the controller, followed by an HCI packet indicator byte, as specified by the UART Transport Layer portion of the most recent Bluetooth Core specification, followed by an HCI packet of the specified packet type, as specified by the Host Controller Interface Functional Specification portion of the most recent Bluetooth Core Specification.
LINKTYPE_AX25_KISS	202	AX.25 packet, with a 1-byte KISS header containing a type indicator.
LINKTYPE_LAPD	203	Link Access Procedures on the D Channel (LAPD) frames, as specified by ITU-T Recommendation Q.920 and ITU-T Recommendation Q.921, starting with the address field, with no pseudo-header.

LINKTYPE_PPP_WITH_DIR	204	PPP, as per RFC 1661 and RFC 1662 , preceded with a one-byte pseudo-header with a zero value meaning received by this host and a non-zero value meaning sent by this host; if the first 2 bytes are 0xff and 0x03, it's PPP in HDLC-like framing, with the PPP header following those two bytes, otherwise it's PPP without framing, and the packet begins with the PPP header. The data in the frame is not octet-stuffed or bit-stuffed.
LINKTYPE_C_HDLC_WITH_DIR	205	Cisco PPP with HDLC framing, as per section 4.3.1 of RFC 1547 , preceded with a one-byte pseudo-header with a zero value meaning received by this host and a non-zero value meaning sent by this host.
LINKTYPE_FRELAY_WITH_DIR	206	Frame Relay LAPF frames, beginning with a one-byte pseudo-header with a zero value meaning received by this host (DCE->DTE) and a non-zero value meaning sent by this host (DTE->DCE), followed by an ITU-T Recommendation Q.922

		LAPF header starting with the address field, and without an FCS at the end of the frame.
LINKTYPE_LAPB_WITH_DIR	207	Link Access Procedure, Balanced (LAPB), as specified by ITU-T Recommendation X.25, preceded with a one-byte pseudo-header with a zero value meaning received by this host (DCE->DTE) and a non-zero value meaning sent by this host (DTE->DCE).
Reserved	208	Reserved for an unspecified link-layer type
LINKTYPE_IPMB_LINUX	209	IPMB over an I2C circuit, with a Linux-specific pseudo-header
LINKTYPE_FLEXRAY	210	Reserved for FlexRay automotive bus
LINKTYPE_MOST	211	Reserved for Media Oriented Systems Transport (MOST) bus
LINKTYPE_LIN	212	Reserved for Local Interconnect Network (LIN) bus for vehicle networks
LINKTYPE_X2E_SERIAL	213	Reserved for X2E serial line captures
LINKTYPE_X2E_XORAYA	214	Reserved for X2E Xoraya data loggers

LINKTYPE_IEEE802_15_4_NONASK_PHY	215	IEEE 802.15.4 Low-Rate Wireless Networks, with each packet having the FCS at the end of the frame, and with the PHY-level data for the O-QPSK, BPSK, GFSK, MSK, and RCC DSS BPSK PHYs (4 octets of 0 as preamble, one octet of SFD, one octet of frame length + reserved bit) preceding the MAC-layer data (starting with the frame control field).
LINKTYPE_LINUX_EVDEV	216	Reserved for Linux evdev messages
LINKTYPE_GSMTAP_UM	217	Reserved for GSM Um interface, with gsmtap header
LINKTYPE_GSMTAP_ABIS	218	Reserved for GSM Abis interface, with gsmtap header
LINKTYPE_MPLS	219	MPLS packets with MPLS label as the header
LINKTYPE_USB_LINUX_MMAPPED	220	USB packets, beginning with a Linux USB header, as specified by the struct usbmon_packet in the Documentation/usb/usbmon.txt file in the Linux source tree. All 64 bytes of the header are present. All fields in the header are in

		host byte order. When performing a live capture, the host byte order is the byte order of the machine on which the packets are captured. When reading a pcap file, the byte order is the byte order for the file, as specified by the file's magic number; when reading a pcapng file, the byte order is the byte order for the section of the pcapng file, as specified by the Section Header Block. For isochronous transfers, the ndesc field specifies the number of isochronous descriptors that follow.
LINKTYPE_DECT	221	Reserved for DECT packets, with a pseudo-header
LINKTYPE_AOS	222	Reserved for OS Space Data Link Protocol
LINKTYPE_WIHART	223	Reserved for Wireless HART (Highway Addressable Remote Transducer)
LINKTYPE_FC_2	224	Fibre Channel FC-2 frames, beginning with a Frame_Header.

LINKTYPE_FC_2_WITH_FRAME_DELIMS	225	Fibre Channel FC-2 frames, beginning an encoding of the SOF, followed by a Frame_Header, and ending with an encoding of the SOF. The encodings represent the frame delimiters as 4-byte sequences representing the corresponding ordered sets, with K28.5 represented as 0xBC, and the D symbols as the corresponding byte values; for example, SOFi2, which is K28.5 - D21.5 - D1.2 - D21.2, is represented as 0xBC 0xB5 0x55 0x55.
LINKTYPE_IPNET	226	Solaris ipnet pseudo-header , followed by an IPv4 or IPv6 datagram.
LINKTYPE_CAN_SOCKETCAN	227	CAN (Controller Area Network) frames, with a pseudo-header followed by the frame payload.
LINKTYPE_IPV4	228	Raw IPv4; the packet begins with an IPv4 header.
LINKTYPE_IPV6	229	Raw IPv6; the packet begins with an IPv6 header.
LINKTYPE_IEEE802_15_4_NOFCS	230	IEEE 802.15.4 Low-Rate Wireless Network, without the FCS at the end of

		the frame.
LINKTYPE_DBUS	231	Raw D-Bus messages , starting with the endianness flag, followed by the message type, etc., but without the authentication handshake before the message sequence.
LINKTYPE_JUNIPER_VS	232	Reserved for Juniper Networks
LINKTYPE_JUNIPER_SRX_E2E	233	Reserved for Juniper Networks
LINKTYPE_JUNIPER_FIBRECHANNEL	234	Reserved for Juniper Networks
LINKTYPE_DVB_CI	235	DVB-CI (DVB Common Interface for communication between a PC Card module and a DVB receiver), with the message format specified by the PCAP format for DVB-CI specification
LINKTYPE_MUX27010	236	Variant of 3GPP TS 27.010 multiplexing protocol (similar to, but not the same as, 27.010).
LINKTYPE_STANAG_5066_D_PDU	237	D_PDUs as described by NATO standard STANAG 5066, starting with the synchronization sequence, and including both header and data CRCs. The current version of STANAG

		5066 is backwards-compatible with the 1.0.2 version , although newer versions are classified.
LINKTYPE_JUNIPER_ATM_CEMIC	238	Reserved for Juniper Networks
LINKTYPE_NFLOG	239	Linux netlink NETLINK NFLOG socket log messages.
LINKTYPE_NETANALYZER	240	Pseudo-header for Hilscher Gesellschaft fuer Systemautomation mbH netANALYZER devices , followed by an Ethernet frame, beginning with the MAC header and ending with the FCS.
LINKTYPE_NETANALYZER_TRANSPARENT	241	Pseudo-header for Hilscher Gesellschaft fuer Systemautomation mbH netANALYZER devices , followed by an Ethernet frame, beginning with the preamble, SFD, and MAC header, and ending with the FCS.
LINKTYPE_IPOIB	242	IP-over-InfiniBand, as specified by RFC 4391 section 6
LINKTYPE_MPEG_2_TS	243	MPEG-2 Transport Stream transport packets, as specified by ISO 13818-1/ ITU-T Recommendation H.222.0 (see table

		2-2 of section 2.4.3.2 Transport Stream packet layer).
LINKTYPE_NG40	244	Pseudo-header for ng4T GmbH's UMTS Iub/Iur-over-ATM and Iub/Iur-over-IP format as used by their ng40 protocol tester , followed by frames for the Frame Protocol as specified by 3GPP TS 25.427 for dedicated channels and 3GPP TS 25.435 for common/shared channels in the case of ATM AAL2 or UDP traffic, by SSCOP packets as specified by ITU-T Recommendation Q.2110 for ATM AAL5 traffic, and by NBAP packets for SCTP traffic.
LINKTYPE_NFC_LLCP	245	Pseudo-header for NFC LLCP packet captures , followed by frame data for the LLCP Protocol as specified by NFCForum-TS-LLCP_1.1
LINKTYPE_PFSYNC	246	Reserved for pfsync output
LINKTYPE_INFINIBAND	247	Raw InfiniBand frames, starting with the Local Routing Header, as specified in Chapter 5 Data packet format of InfiniBand[TM] Architectural

		Specification Release 1.2.1 Volume 1 - General Specifications
LINKTYPE_SCTP	248	SCTP packets, as defined by RFC 4960 , with no lower- level protocols such as IPv4 or IPv6.
LINKTYPE_USBPCAP	249	USB packets, beginning with a USBPcap header
LINKTYPE_RTAC_SERIAL	250	Serial-line packet header for the Schweitzer Engineering Laboratories RTAC product , followed by a payload for one of a number of industrial control protocols.
LINKTYPE_BLUETOOTH_LE_LL	251	Bluetooth Low Energy air interface Link Layer packets, in the format described in section 2.1 PACKET FORMAT of volume 6 of the Bluetooth Specification Version 4.0 (see PDF page 2200), but without the Preamble.
LINKTYPE_WIRESHARK_UPPER_PDU	252	Reserved for Wireshark
LINKTYPE_NETLINK	253	Linux Netlink capture encapsulation
LINKTYPE_BLUETOOTH_LINUX_MONITOR	254	Bluetooth Linux

		Monitor encapsulation of traffic for the BlueZ stack
LINKTYPE_BLUETOOTH_BREDR_BB	255	Bluetooth Basic Rate and Enhanced Data Rate baseband packets
LINKTYPE_BLUETOOTH_LE_LL_WITH_PHDR	256	Bluetooth Low Energy link-layer packets
LINKTYPE_PROFIBUS_DL	257	PROFIBUS data link layer packets, as specified by IEC standard 61158-4-3, beginning with the start delimiter, ending with the end delimiter, and including all octets between them.
LINKTYPE_PKTAP	258	Apple PKTAP capture encapsulation
LINKTYPE_EPON	259	Ethernet-over-passive-optical-network packets, starting with the last 6 octets of the modified preamble as specified by 65.1.3.2 Transmit in Clause 65 of Section 5 of IEEE 802.3 , followed immediately by an Ethernet frame.
LINKTYPE_IPMI_HPM_2	260	IPMI trace packets, as specified by Table 3-20 Trace Data Block Format in the PICMG HPM.2 specification The time stamps for

		packets in this format must match the time stamps in the Trace Data Blocks.
LINKTYPE_ZWAVE_R1_R2	261	Z-Wave RF profile R1 and R2 packets , as specified by ITU-T Recommendation G.9959 , with some MAC layer fields moved.
LINKTYPE_ZWAVE_R3	262	Z-Wave RF profile R3 packets , as specified by ITU-T Recommendation G.9959 , with some MAC layer fields moved.
LINKTYPE_WATTSTOPPER_DLM	263	Formats for WattStopper Digital Lighting Management (DLM) and Legrand Nitoo Open protocol common packet structure captures.
LINKTYPE_ISO_14443	264	Messages between ISO 14443 contactless smartcards (Proximity Integrated Circuit Card, PICC) and card readers (Proximity Coupling Device, PCD), with the message format specified by the PCAP format for ISO14443 specification
LINKTYPE_RDS	265	Radio data system (RDS) groups, as per IEC 62106,

		encapsulated in this form
LINKTYPE_USB_DARWIN	266	USB packets, beginning with a Darwin (macOS, etc.) USB header
LINKTYPE_OPENFLOW	267	Reserved for OpenBSD DLT_OPENFLOW
LINKTYPE_SDL	268	SDLC packets, as specified by Chapter 1, DLC Links, section Synchronous Data Link Control (SDLC) of Systems Network Architecture Formats, GA27-3136-20 , without the flag fields, zero-bit insertion, or Frame Check Sequence field, containing SNA path information units (PIUs) as the payload.
LINKTYPE_TI_LL_N_SNIFFER	269	Reserved for Texas Instruments protocol sniffer
LINKTYPE_LORATAP	270	LoRaTap pseudo-header , followed by the payload, which is typically the PHYPayload from the LoRaWan specification
LINKTYPE_VSOCK	271	Protocol for communication between host and guest machines in VMware and KVM hypervisors.

LINKTYPE_NORDIC_BLE	272	Messages to and from a Nordic Semiconductor nRF Sniffer for Bluetooth LE packets, beginning with a pseudo-header
LINKTYPE_DOCSIS31_XRA31	273	DOCSIS packets and bursts, preceded by a pseudo-header giving metadata about the packet
LINKTYPE_ETHERNET_MPACKET	274	mPackets, as specified by IEEE 802.3br Figure 99-4, starting with the preamble and always ending with a CRC field.
LINKTYPE_DISPLAYPORT_AUX	275	DisplayPort AUX channel monitoring data as specified by VESA DisplayPort (DP) Standard preceded by a pseudo-header
LINKTYPE_LINUX_SLL2	276	Linux cooked capture encapsulation v2
LINKTYPE_SERCOS_MONITOR	277	Reserved for Sercos Monitor
LINKTYPE_OPENVIZSLA	278	Openvizsla FPGA-based USB sniffer
LINKTYPE_EBHSCR	279	Elektrobit High Speed Capture and Replay (EBHSCR) format
LINKTYPE_VPP_DISPATCH	280	Records in traces from the http://fd.io VPP graph dispatch tracer, in the the

		graph dispatcher trace format
LINKTYPE_DSA_TAG_BRCM	281	Ethernet frames, with a switch tag inserted between the source address field and the type/length field in the Ethernet header.
LINKTYPE_DSA_TAG_BRCM_PREPEND	282	Ethernet frames, with a switch tag inserted before the destination address in the Ethernet header.
LINKTYPE_IEEE802_15_4_TAP	283	IEEE 802.15.4 Low- Rate Wireless Networks, with a pseudo-header containing TLVs with metadata preceding the 802.15.4 header.
LINKTYPE_DSA_TAG_DSA	284	Ethernet frames, with a switch tag inserted between the source address field and the type/length field in the Ethernet header.
LINKTYPE_DSA_TAG_EDSA	285	Ethernet frames, with a programmable Ethernet type switch tag inserted between the source address field and the type/ length field in the Ethernet header.
LINKTYPE_ELEE	286	Payload of lawful intercept packets using the ELEE protocol The packet begins with the ELEE

		header; it does not include any transport-layer or lower-layer headers for protocols used to transport ELEE packets.
LINKTYPE_Z_WAVE_SERIAL	287	Serial frames transmitted between a host and a Z-Wave chip over an RS-232 or USB serial connection, as described in section 5 of the Z-Wave Serial API Host Application Programming Guide
LINKTYPE_USB_2_0	288	USB 2.0, 1.1, or 1.0 packet, beginning with a PID, as described by Chapter 8 Protocol Layer of the the Universal Serial Bus Specification Revision 2.0
LINKTYPE_ATSC_ALP	289	ATSC Link-Layer Protocol frames, as described in section 5 of the A/330 Link-Layer Protocol specification, found at the ATSC 3.0 standards page , beginning with a Base Header

Table 1

9. Contributors

[Insert pcap developers etc. here].

10. Acknowledgments

The authors wish to thank [insert list here] and many others for their invaluable comments.

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/rfc/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/rfc/rfc8174>>.

11.2. Informative References

- [I-D.tuexen-opsawg-pcapng] Tuexen, M., Risso, F., Bongertz, J., Combs, G., Harris, G., Chaudron, E., and M. C. Richardson, "PCAP Next Generation (pcapng) Capture File Format", Work in Progress, Internet-Draft, draft-tuexen-opsawg-pcapng-04, 4 October 2021, <<https://datatracker.ietf.org/doc/html/draft-tuexen-opsawg-pcapng-04>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/rfc/rfc8126>>.
- [Radiotap] radiotap.org, "Radiotap Web site", n.d., <<http://www.radiotap.org/>>.
- [AVS] Peachy, S., "Archived AVS specification", n.d., <<http://web.archive.org/web/20040803232023/http://www.shaftnet.org/~pizza/software/capturefrm.txt>>.

Authors' Addresses

Guy Harris (editor)

Email: gharris@sonic.net

Michael C. Richardson
Sandelman Software Works Inc

Email: mcr+ietf@sandelman.ca
URI: <http://www.sandelman.ca/>

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 7 September 2022

E. Lear
Cisco Systems
S. Rose
NIST
6 March 2022

Discovering and Retrieving Software Transparency and Vulnerability
Information
draft-ietf-opsawg-sbom-access-05

Abstract

To improve cybersecurity posture, automation is necessary to locate what software is running on a device, whether that software has known vulnerabilities, and what, if any recommendations suppliers may have. This memo specifies a model to provide access to this information. It may optionally be discovered through manufacturer usage descriptions.

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 7 September 2022.

Copyright Notice

Copyright (c) 2022 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	2
1.1. How This Information Is Retrieved	5
1.2. Formats	5
1.3. Discussion points	5
2. The well-known transparency endpoint set	6
3. The mud-transparency extension model extension	6
4. The mud-sbom augmentation to the MUD YANG model	6
5. Examples	10
5.1. Without ACLS	10
5.2. SBOM Located on the Device	12
5.3. Further contact required.	13
5.4. With ACLS	14
6. Security Considerations	16
7. IANA Considerations	18
7.1. MUD Extension	18
7.2. YANG Registration	18
7.3. Well-Known Prefix	18
8. Acknowledgments	19
9. References	19
9.1. Normative References	19
9.2. Informative References	20
Appendix A. Changes from Earlier Versions	20
Authors' Addresses	21

1. Introduction

A number of activities have been working to improve visibility to what software is running on a system, and what vulnerabilities that software may have[EO2021].

Put simply, we seek to answer two classes of questions *at scale*:

- * Is this system vulnerable to a particular vulnerability?
- * Which devices in a particular environment contain vulnerabilities that require some action?

This memo doesn't specify the format of this information, but rather only how to locate and retrieve these objects.

Software bills of materials (SBOMs) are descriptions of what software, including versioning and dependencies, a device contains. There are different SBOM formats such as Software Package Data Exchange [SPDX] or CycloneDX[CycloneDX12].

System vulnerabilities may similarly be described using several data formats, including the aforementioned CycloneDX, Common Vulnerability Reporting Framework [CVRF], the Common Security Advisory Format [CSAF]. This information is typically used to report to customers the state of a system.

These two classes of information can be used in concert. For instance, a network management tool may discover that a system makes use of a particular software component that has a known vulnerability, and a vulnerability report may be used to indicate what if any versions of software correct that vulnerability, or whether the system exercises the vulnerable code at all.

Both classes of information elements are optional under the model specified in this memo. One can provide only an SBOM, only vulnerability information, or both an SBOM and vulnerability information.

Note that SBOM formats may also carry other information, the most common being any licensing terms. Because this specification is neutral regarding content, it is left for format developers such as the Linux Foundation, OASIS, and ISO to decide what attributes they will support.

This memo does not specify how vulnerability information may be retrieved directly from the endpoint. That's because vulnerability information changes occur at different rates to software updates. However, some SBOM formats may also contain vulnerability information.

SBOMs and vulnerability information are advertised and retrieved through the use of a YANG augmentation of the Manufacturer User Description (MUD) model [RFC8520]. Note that the schema creates a grouping that can also be used independently of MUD. Moreover, other MUD features, such as access controls, needn't be present.

The mechanisms specified in this document are meant to satisfy several use cases:

- * A network-layer management system retrieving information from an IoT device as part of its ongoing lifecycle. Such devices may or may not have query interfaces available.
- * An application-layer management system retrieving vulnerability or SBOM information in order to evaluate the posture of an application server of some form. These application servers may themselves be containers or hypervisors. Discovery of the topology of a server is beyond the scope of this memo.

To satisfy these two key use cases, objects may be found in one of three ways:

- * on devices themselves
- * on a web site (e.g., via URI)
- * through some form of out-of-band contact with the supplier.

In the first case, devices will have interfaces that permit direct retrieval. Examples of these interfaces might be an HTTP, COAP or [OpenC2] endpoint for retrieval. There may also be private interfaces as well.

In the second case, when a device does not have an appropriate retrieval interface, but one is directly available from the manufacturer, a URI to that information **MUST** be discovered.

In the third case, a supplier may wish to make an SBOM or vulnerability information available under certain circumstances, and may need to individually evaluate requests. The result of that evaluation might be the SBOM or vulnerability itself or a restricted URL or no access.

To enable application-layer discovery, this memo defines a well-known URI [RFC8615]. Management or orchestration tools can query this well-known URI to retrieve a system's SBOM or vulnerability information. Further queries may be necessary based on the content and structure of the response.

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.

1.1. How This Information Is Retrieved

For devices that can emit a URL or can establish a well-known URI, the mechanism may be highly automated. For devices that have a URL in either their documentation or within a QR code on a box, the mechanism is semi-automated (someone has to scan the QR code or enter the URL).

Note that vulnerability and SBOM information is likely to change at different rates. The MUD semantics provide a way for manufacturers to control how often tooling should check for those changes through the cache-validity node.

1.2. Formats

There are multiple ways to express both SBOMs and vulnerability information. When these are retrieved either directly from the device or directly from a web server, tools will need to observe the content-type header to determine precisely which format is being transmitted. Because IoT devices in particular have limited capabilities, use of a specific Accept: header in HTTP or the Accept Option in CoAP is NOT RECOMMENDED. Instead, backend tooling is encouraged to support all known formats, and SHOULD silently discard SBOM information sent with a media type that is not understood.

Some formats may support both vulnerability and software inventory information. When both vulnerability and software inventory information is available from the same location, both sbom and vuln nodes MUST indicate that. Network management systems retrieving this information MUST take note that the identical resource is being retrieved rather than retrieving it twice.

1.3. Discussion points

The following is discussion to be removed at time of RFC publication.

- * Is the model structured correctly?
- * Are there other retrieval mechanisms that need to be specified?
- * Do we need to be more specific in how to authenticate and retrieve SBOMs?
- * What are the implications if the MUD URL is an extension in a certificate (e.g. an IDevID cert)?

2. The well-known transparency endpoint set

Two well known endpoints are defined:

- * `"/.well-known/sbom"` retrieves an SBOM.
- * `"/.well-known/openc2"` is the HTTPS binding to OpenC2.

As discussed previously, the precise format of a response is based on the Content-type provided.

3. The mud-transparency extension model extension

We now formally define this extension. This is done in two parts. First, the extension name "transparency" is listed in the "extensions" array of the MUD file. N.B., this schema extension is intended to be used wherever it might be appropriate (e.g., not just MUD).

Second, the "mud" container is augmented with a list of SBOM sources.

This is done as follows:

module: ietf-mud-transparency

```
augment /mud:mud:
  +--rw transparency
    +--rw (sbom-retrieval-method)?
      +--:(cloud)
        +--rw sboms* [version-info]
          +--rw version-info      string
          +--rw sbom-url?         inet:uri
      +--:(local-well-known)
        +--rw sbom-local-well-known? enumeration
      +--:(sbom-contact-info)
        +--rw sbom-contact-uri      inet:uri
    +--rw (vuln-retrieval-method)?
      +--:(cloud)
        +--rw vuln-url?             inet:uri
      +--:(vuln-contact-info)
        +--rw contact-uri           inet:uri
```

4. The mud-sbom augmentation to the MUD YANG model

```
<CODE BEGINS>
file "ietf-mud-transparency@2021-10-22.yang"
module ietf-mud-transparency {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-mud-transparency";
  prefix mudtx;

  import ietf-inet-types {
    prefix inet;
    reference "RFC 6991";
  }
  import ietf-mud {
    prefix mud;
    reference "RFC 8520";
  }

  organization
    "IETF OPSAWG (Ops Area) Working Group";
  contact
    "WG Web: http://tools.ietf.org/wg/opsawg/
    WG List: opsawg@ietf.org

    Editor: Eliot Lear lear@cisco.com
    Editor: Scott Rose scott.rose@nist.gov";
  description
    "This YANG module augments the ietf-mud model to provide for
    reporting of SBOMs and vulnerability information."

  Copyright (c) 2020 IETF Trust and the persons identified as
  authors of the code. All rights reserved.

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

  This version of this YANG module is part of RFC XXXX
  (https://www.rfc-editor.org/info/rfcXXXX);
  see the RFC itself for full legal notices.

  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. ";
```

```
revision 2021-07-06 {
  description
    "Initial proposed standard.";
  reference
    "RFC XXXX: Discovering and Retrieving Software Transparency
    and Vulnerability Information";
}

grouping transparency-extension {
  description
    "Transparency extension grouping";
  container transparency {
    description
      "container of methods to get an SBOM.";
    choice sbom-retrieval-method {
      description
        "How to find SBOM information";
      case cloud {
        list sboms {
          key "version-info";
          description
            "A list of SBOMs tied to different s/w
            or h/w versions.";
          leaf version-info {
            type string;
            description
              "The version to which this SBOM refers.";
          }
          leaf sbom-url {
            type inet:uri;
            description
              "A statically located URI.";
          }
        }
      }
      case local-well-known {
        leaf sbom-local-well-known {
          type enumeration {
            enum http {
              description
                "Use http (insecure) to retrieve
                SBOM information. This method is NOT RECOMMENDED,
                but may be unavoidable for certain classes of
                deployment, where TLS has not or cannot be implemented";
            }
            enum https {
              description
                "Use https (secure) to retrieve SBOM information.";
            }
          }
        }
      }
    }
  }
}
```

```
    }
    enum coap {
      description
        "Use COAP (insecure) to retrieve SBOM. This method
        is NOT RECOMMENDED, although it may be unavoidable
        for certain classes of implementations/deployments.";
    }
    enum coaps {
      description
        "Use COAPS (secure) to retrieve SBOM";
    }
    enum openc2 {
      description
        "Use OpenC2 endpoint.
        This is https://{host}/.well-known/openc2";
    }
  }
  description
    "Which communication protocol to choose.";
}
}
case sbom-contact-info {
  leaf sbom-contact-uri {
    type inet:uri;
    mandatory true;
    description
      "This MUST be either a tel, http, https, or
      mailto uri schema that customers can use to
      contact someone for SBOM information.";
  }
}
}
choice vuln-retrieval-method {
  description
    "How to find vulnerability information";
  case cloud {
    leaf vuln-url {
      type inet:uri;
      description
        "A statically located URL.";
    }
  }
}
case vuln-contact-info {
  leaf contact-uri {
    type inet:uri;
    mandatory true;
    description
      "This MUST be either a tel, http, https, or
```

```
        mailto uri schema that customers can use to
        contact someone for vulnerability information.";
    }
  }
}

augment "/mud:mud" {
  description
    "Add extension for software transparency.";
  uses transparency-extension;
}
}
<CODE ENDS>
```

5. Examples

In this example MUD file that uses a cloud service, the modelX presents a location of the SBOM in a URL. Note, the ACLs in a MUD file are NOT required, although they are a very good idea for IP-based devices.

5.1. Without ACLS

This first MUD file demonstrates how to get SBOM and vulnerability information without ACLs.

```
{
  "ietf-mud:mud": {
    "mud-version": 1,
    "extensions": [
      "ol",
      "transparency"
    ],
    "ol": {
      "owners": [
        "Copyright (c) Example, Inc. 2022. All Rights Reserved"
      ],
      "spdx-tag": "0BSD"
    },
    "mudtx:transparency": {
      "sbom-local-well-known": "https",
      "vuln-url": "https://iot.example.com/info/modelX/csaf.json"
    },
    "mud-url": "https://iot.example.com/modelX.json",
    "mud-signature": "https://iot.example.com/modelX.p7s",
    "last-update": "2022-01-05T13:29:12+00:00",
    "cache-validity": 48,
    "is-supported": true,
    "systeminfo": "retrieving vuln and SBOM info via a cloud service",
    "mfg-name": "Example, Inc.",
    "documentation": "https://iot.example.com/doc/modelX",
    "model-name": "modelX"
  }
}
```

The second example demonstrates that just SBOM information is included.

```
{
  "ietf-mud:mud": {
    "mud-version": 1,
    "extensions": [
      "ol",
      "transparency"
    ],
    "ol": {
      "owners": [
        "Copyright (c) Example, Inc. 2022. All Rights Reserved"
      ],
      "spdx-tag": "0BSD"
    },
    "mudtx:transparency": {
      "sbom-local-well-known": "https"
    },
    "mud-url": "https://iot.example.com/modelX.json",
    "mud-signature": "https://iot.example.com/modelX.p7s",
    "last-update": "2022-01-05T13:29:47+00:00",
    "cache-validity": 48,
    "is-supported": true,
    "systeminfo": "retrieving vuln and SBOM info via a cloud service",
    "mfg-name": "Example, Inc.",
    "documentation": "https://iot.example.com/doc/modelX",
    "model-name": "modelX"
  }
}
```

5.2. SBOM Located on the Device

In this example, the SBOM is retrieved from the device, while vulnerability information is available from the cloud. This is likely a common case, because vendors may learn of vulnerability information more frequently than they update software.

```
{
  "ietf-mud:mud": {
    "mud-version": 1,
    "extensions": [
      "transparency"
    ],
    "mudtx:transparency": {
      "sbom-local-well-known": "https",
      "vuln-url": "https://iot-device.example.com/info/modelX/csaf.json"
    },
    "mud-url": "https://iot-device.example.com/modelX.json",
    "mud-signature": "https://iot-device.example.com/modelX.p7s",
    "last-update": "2022-01-05T13:25:14+00:00",
    "cache-validity": 48,
    "is-supported": true,
    "systeminfo": "retrieving vuln and SBOM info via a cloud service",
    "mfg-name": "Example, Inc.",
    "documentation": "https://iot-device.example.com/doc/modelX",
    "model-name": "modelX"
  }
}
```

5.3. Further contact required.

In this example, the network manager must take further steps to retrieve SBOM information. Vulnerability information is still available.


```
{
  "ietf-mud:mud": {
    "mud-version": 1,
    "extensions": [
      "transparency"
    ],
    "ietf-mud-transparency:transparency": {
      "contact-info": "https://iot-device.example.com/contact-info.html",
      "vuln-url": "https://iot-device.example.com/info/modelX/csaf.json"
    },
    "mud-url": "https://iot-device.example.com/modelX.json",
    "mud-signature": "https://iot-device.example.com/modelX.p7s",
    "last-update": "2021-07-09T06:16:42+00:00",
    "cache-validity": 48,
    "is-supported": true,
    "systeminfo": "retrieving vuln and SBOM info via a cloud service",
    "mfg-name": "Example, Inc.",
    "documentation": "https://iot-device.example.com/doc/modelX",
    "model-name": "modelX"
  }
}
```

5.4. With ACLS

Finally, here is a complete example where the device provides SBOM and vulnerability information, as well as access-control information.

```
{
  "ietf-mud:mud": {
    "mud-version": 1,
    "extensions": [
      "ol",
      "transparency"
    ],
    "ol": {
      "owners": [
        "Copyright (c) Example, Inc. 2022. All Rights Reserved"
      ],
      "spdx-tag": "0BSD"
    },
    "mudtx:transparency": {
      "sbom-local-well-known": "https",
      "vuln-url": "https://iot.example.com/info/modelX/csaf.json"
    },
    "mud-url": "https://iot.example.com/modelX.json",
    "mud-signature": "https://iot.example.com/modelX.p7s",
    "last-update": "2022-01-05T13:30:31+00:00",
    "cache-validity": 48,
  }
}
```

```
"is-supported": true,
"systeminfo": "retrieving vuln and SBOM info via a cloud service",
"mfg-name": "Example, Inc.",
"documentation": "https://iot.example.com/doc/modelX",
"model-name": "modelX",
"from-device-policy": {
  "access-lists": {
    "access-list": [
      {
        "name": "mud-65443-v4fr"
      }
    ]
  }
},
"to-device-policy": {
  "access-lists": {
    "access-list": [
      {
        "name": "mud-65443-v4to"
      }
    ]
  }
},
"ietf-access-control-list:acls": {
  "acl": [
    {
      "name": "mud-65443-v4to",
      "type": "ipv4-acl-type",
      "aces": {
        "ace": [
          {
            "name": "cl0-todev",
            "matches": {
              "ipv4": {
                "ietf-acldns:src-dnsname": "iotserver.example.com"
              }
            },
            "actions": {
              "forwarding": "accept"
            }
          }
        ]
      }
    }
  ]
},
{
  "name": "mud-65443-v4fr",
  "type": "ipv4-acl-type",
```

```
"aces": {
  "ace": [
    {
      "name": "cl0-frdev",
      "matches": {
        "ipv4": {
          "ietf-acldns:dst-dnsname": "iotserver.example.com"
        }
      },
      "actions": {
        "forwarding": "accept"
      }
    }
  ]
}
]
```

At this point, the management system can attempt to retrieve the SBOM, and determine which format is in use through the content-type header on the response to a GET request, independently repeat the process for vulnerability information, and apply ACLs, as appropriate.

6. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

N.B., for MUD, the mandatory method of retrieval is TLS.

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). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

The ietf-mud-transparency module has no operational impact on the element itself, and is used to discover state information that may be available on or off the element. In as much as the module itself is made writeable, this only indicates a change in how to retrieve what read-only elements. However, that does not mean there are no risks. These are discussed below, and are applicable to all nodes within the transparency container.

If an attacker modifies the elements, they may misdirect automation to retrieve a different set of URLs than was intended by the designer. This in turn leads to two specific sets of risks:

- * the information retrieved would be false.
- * the URLs themselves point to malware.

To address either risk, any change in a URL, and in particular to the authority section, should be treated with some suspicion. One mitigation would be to test any cloud-based URL against a reputation service.

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. These are the subtrees and data nodes and their sensitivity/vulnerability:

SBOMs provide an inventory of software. If software is available to an attacker, the attacker may well already be able to derive this very same software inventory. Manufacturers MAY restrict access to SBOM information using appropriate authorization semantics within HTTP. In particular, if a system attempts to retrieve an SBOM via HTTP and the client is not authorized, the server MUST produce an appropriate error, with instructions on how to register a particular client. One example may be to issue a certificate to the client for this purpose after a registration process has taken place. Another example would involve the use of OAUTH in combination with a federations of SBOM servers.

Another risk is a skew in the SBOM listing and the actual software inventory of a device/container. For example, a manufacturer may update the SBOM on its server, but an individual device has not been upgraded yet. This may result in an incorrect policy being applied to a device. A unique mapping of a device's software version and its SBOM can minimize this risk.

To further mitigate attacks against a device, manufacturers SHOULD recommend access controls.

Vulnerability information is generally made available to such databases as NIST's National Vulnerability Database. It is possible that vendor may wish to release information early to some customers. We do not discuss here whether that is a good idea, but if it is employed, then appropriate access controls and authorization SHOULD be applied to the vulnerability resource.

7. IANA Considerations

7.1. MUD Extension

The IANA is requested to add "transparency" to the MUD extensions registry as follows:

Extension Name: transparency
Standard reference: This document

7.2. YANG Registration

The following YANG module should be registered in the "YANG Module Names" registry:

Name: ietf-mud
URN: urn:ietf:params:xml:ns:yang:ietf-mud-transparency
Prefix: mudtx
Registrant contact: The IESG
Reference: This memo

7.3. Well-Known Prefix

The following well known URIs are requested in accordance with [RFC8615]:

URI suffix: "sbom"
Change controller: "IETF"
Specification document: This memo
Related information: See ISO/IEC 19970-2 and SPDX.org

URI suffix: "openc2"
Change controller: "IETF"
Specification document: This memo
Related information: OpenC2 Project

8. Acknowledgments

Thanks to Russ Housley, Dick Brooks, Tom Petch, Nicolas Comstedt, who provided review comments.

9. References

9.1. 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>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [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>>.
- [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>>.
- [RFC8520] Lear, E., Droms, R., and D. Romascanu, "Manufacturer Usage Description Specification", RFC 8520, DOI 10.17487/RFC8520, March 2019, <<https://www.rfc-editor.org/info/rfc8520>>.
- [RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, DOI 10.17487/RFC8615, May 2019, <<https://www.rfc-editor.org/info/rfc8615>>.

9.2. Informative References

- [CSAF] OASIS, "Common Security Advisory Format", July 2021, <<https://github.com/oasis-tcs/csaf>>.
- [CVRF] Santos, O., Ed., "Common Vulnerability Reporting Framework (CVRF) Version 1.2", September 2017, <<https://docs.oasis-open.org/csaf/csaf-cvrf/v1.2/csaf-cvrf-v1.2.pdf>>.
- [CycloneDX12] cyclonedx.org, "CycloneDX XML Reference v1.2", May 2020.
- [EO2021] Biden, J., "Executive Order 14028, Improving the Nations Cybersecurity", May 2021.
- [OpenC2] Lemire, D., Ed., "Specification for Transfer of OpenC2 Messages via HTTPS Version 1.0", July 2019, <<https://docs.oasis-open.org/openc2/open-impl-https/v1.0/open-impl-https-v1.0.html>>.
- [SPDX] The Linux Foundation, "SPDX Specification 2.1", 2016.

Appendix A. Changes from Earlier Versions

Draft -04: * Address review comments

Draft -02:

* include vulnerability information

Draft -01:

* some modest changes

Draft -00:

* Initial revision

Authors' Addresses

Eliot Lear
Cisco Systems
Richtistrasse 7
CH-8304 Wallisellen
Switzerland
Phone: +41 44 878 9200
Email: lear@cisco.com

Scott Rose
NIST
100 Bureau Dr
Gaithersburg MD, 20899
United States of America
Phone: +1 301-975-8439
Email: scott.rose@nist.gov

OPSAWG
Internet-Draft
Intended status: Informational
Expires: 8 September 2022

B. Claise
J. Quilbeuf
Huawei
D. Lopez
Telefonica I+D
D. Voyer
Bell Canada
T. Arumugam
Cisco Systems, Inc.
7 March 2022

Service Assurance for Intent-based Networking Architecture
draft-ietf-opsawg-service-assurance-architecture-03

Abstract

This document describes an architecture for Service Assurance for Intent-based Networking (SAIN). This architecture aims at assuring that service instances are running as expected. As services rely upon multiple sub-services provided by the underlying network devices and functions, getting the assurance of a healthy service is only possible with a holistic view of all involved elements. This architecture not only helps to correlate the service degradation with the network root cause but also the impacted services when a network component fails or degrades.

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 8 September 2022.

Copyright Notice

Copyright (c) 2022 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. Terminology	2
2. Introduction	5
3. Architecture	7
3.1. Inferring a Service Instance Configuration into an Assurance Graph	10
3.1.1. Circular Dependencies	12
3.2. Intent and Assurance Graph	16
3.3. Subservices	16
3.4. Building the Expression Graph from the Assurance Graph	17
3.5. Building the Expression from a Subservice	18
3.6. Open Interfaces with YANG Modules	18
3.7. Handling Maintenance Windows	18
3.8. Flexible Architecture	19
3.9. Timing	20
3.10. New Assurance Graph Generation	21
4. Security Considerations	21
5. IANA Considerations	22
6. Contributors	22
7. Open Issues	22
8. References	22
8.1. Normative References	22
8.2. Informative References	22
Appendix A. Changes between revisions	24
Acknowledgements	25
Authors' Addresses	25

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.

SAIN agent: A functional component that communicates with a device, a set of devices, or another agent to build an expression graph from a received assurance graph and perform the corresponding computation of the health status and symptoms.

Assurance case: According to [Piovesan2017]: "An assurance case is a structured argument, supported by evidence, intended to justify that a system is acceptably assured relative to a concern (such as safety or security) in the intended operating environment."

Assurance graph: A Directed Acyclic Graph (DAG) representing the assurance case for one or several service instances. The nodes (also known as vertices in the context of DAG) are the service instances themselves and the subservices, the edges indicate a dependency relations.

SAIN collector: A functional component that fetches or receives the computer-consumable output of the SAIN agent(s) and displays it in a user friendly form or process it locally.

DAG: Directed Acyclic Graph.

ECMP: Equal Cost Multiple Paths

Expression graph: A generic term for a DAG representing a computation in SAIN. More specific terms are:

- * **Subservice expressions:** Is an expression graph representing all the computations to execute for a subservice.
- * **Service expressions:** Is an expression graph representing all the computations to execute for a service instance, i.e., including the computations for all dependent subservices.
- * **Global computation graph:** Is an expression graph representing all the computations to execute for all services instances (i.e., all computations performed).

Dependency: The directed relationship between subservice instances in the assurance graph.

Informational Dependency: Type of dependency whose health score does not impact the health score of its parent subservice or service instance(s) in the assurance graph. However, the symptoms should be taken into account in the parent service instance or subservice instance(s), for informational reasons.

Impacting Dependency: Type of dependency whose score impacts the score of its parent subservice or service instance(s) in the assurance graph. The symptoms are taken into account in the parent service instance or subservice instance(s), as the impacting reasons.

Metric: An information retrieved from the network running the assured service.

Metric engine: A functional components that maps metrics to a list of candidate metric implementations depending on the network element.

Metric implementation: Actual way of retrieving a metric from a network element.

Network service YANG module: describes the characteristics of a service as agreed upon with consumers of that service [RFC8199].

Service instance: A specific instance of a service.

Service configuration orchestrator: Quoting RFC8199, "Network Service YANG Modules describe the characteristics of a service, as agreed upon with consumers of that service. That is, a service module does not expose the detailed configuration parameters of all participating network elements and features but describes an abstract model that allows instances of the service to be decomposed into instance data according to the Network Element YANG Modules of the participating network elements. The service-to-element decomposition is a separate process; the details depend on how the network operator chooses to realize the service. For the purpose of this document, the term "orchestrator" is used to describe a system implementing such a process."

SAIN orchestrator: A functional component that is in charge of fetching the configuration specific to each service instance and converting it into an assurance graph.

Health status: Score and symptoms indicating whether a service instance or a subservice is "healthy". A non-maximal score must always be explained by one or more symptoms.

Health score: Integer ranging from 0 to 100 indicating the health of a subservice. A score of 0 means that the subservice is broken, a score of 100 means that the subservice in question is operating as expected.

Subservice: Part or functionality of the network system that can be independently assured as a single entity in assurance graph.

Strongly connected component: subset of a directed graph such that there is a (directed) path from any node of the subset to any other node. A DAG does not contain any strongly connected component.

Symptom: Reason explaining why a service instance or a subservice is not completely healthy.

2. Introduction

Network Service YANG Modules [RFC8199] describe the configuration, state data, operations, and notifications of abstract representations of services implemented on one or multiple network elements.

Quoting RFC8199: "Network Service YANG Modules describe the characteristics of a service, as agreed upon with consumers of that service. That is, a service module does not expose the detailed configuration parameters of all participating network elements and features but describes an abstract model that allows instances of the service to be decomposed into instance data according to the Network Element YANG Modules of the participating network elements. The service-to-element decomposition is a separate process; the details depend on how the network operator chooses to realize the service. For the purpose of this document, the term "orchestrator" is used to describe a system implementing such a process."

Service configuration orchestrators deploy Network Service YANG Modules [RFC8199] that will infer network-wide configuration and, therefore the configuration of the appropriate device modules (Section 3 of [RFC8969]). Network configuration is based on these device YANG modules, with protocol/encoding such as NETCONF/XML [RFC6241], RESTCONF/JSON [RFC8040], gNMI/gRPC/protobuf, etc. Knowing that a configuration is applied doesn't imply that the service is running as expected (e.g., the service might be degraded because of a failure in the network), the network operator must monitor the service operational data at the same time as the configuration (Section 3.3 of [RFC8969]). The industry has been standardizing on telemetry to push network element performance information.

A network administrator needs to monitor her network and services as a whole, independently of the use cases or the management protocols. With different protocols come different data models, and different ways to model the same type of information. When network administrators deal with multiple protocols, the network management must perform the difficult and time-consuming job of mapping data models: the model used for configuration with the model used for monitoring. This problem is compounded by a large, disparate set of data sources (MIB modules, YANG models [RFC7950], IPFIX information

elements [RFC7011], syslog plain text [RFC3164], TACACS+ [RFC8907], RADIUS [RFC2865], etc.). In order to avoid this data model mapping, the industry converged on model-driven telemetry to stream the service operational data, reusing the YANG models used for configuration. Model-driven telemetry greatly facilitates the notion of closed-loop automation whereby events/status from the network drive remediation changes back into the network.

However, it proves difficult for network operators to correlate the service degradation with the network root cause. For example, why does my L3VPN fail to connect? Why is this specific service slow? The reverse, i.e., which services are impacted when a network component fails or degrades, is even more interesting for the operators. For example, which services are impacted when this specific optic dBm begins to degrade? Which applications are impacted by this ECMP imbalance? Is that issue actually impacting any other customers?

Intent-based approaches are often declarative, starting from a statement of "The service works as expected" and trying to enforce it. Such approaches are mainly suited for greenfield deployments.

Aligned with Section 3.3 of [RFC7149], and instead of approaching intent from a declarative way, this architecture focuses on already defined services and tries to infer the meaning of "The service works as expected". To do so, the architecture works from an assurance graph, deduced from the service definition and from the network configuration. In some cases, the assurance graph may also be explicitly completed to add an intent not exposed in the service model itself (e.g. the service must rely on a backup physical path). This assurance graph is decomposed into components, which are then assured independently. The root of the assurance graph represents the service to assure, and its children represent components identified as its direct dependencies; each component can have dependencies as well. The SAIN architecture updates the assurance graph when services are modified or when the network conditions change.

When a service is degraded, the SAIN architecture will highlight, to the best of its knowledge, where in the assurance service graph to look, as opposed to going hop by hop to troubleshoot the issue. Not only can this architecture help to correlate service degradation with network root cause/symptoms, but it can deduce from the assurance graph the number and type of services impacted by a component degradation/failure. This added value informs the operational team where to focus its attention for maximum return. Indeed, the operational team should focus his priority on the degrading/failing components impacting the highest number customers, especially the ones with the SLA contracts involving penalties in case of failure.

This architecture provides the building blocks to assure both physical and virtual entities and is flexible with respect to services and subservices, of (distributed) graphs, and of components (Section 3.8).

3. Architecture

The goal of SAIN is to assure that service instances are operating correctly and if not, to pinpoint what is wrong. More precisely, SAIN computes a score for each service instance and outputs symptoms explaining that score, especially why the score is not maximal. The score augmented with the symptoms is called the health status.

The SAIN architecture is a generic architecture, applicable to multiple environments. Obviously wireline but also wireless, but also different domains such as 5G, NFV domain with a virtual infrastructure manager (VIM), etc. And as already noted, for physical or virtual devices, as well as virtual functions. Thanks to the distributed graph design principle, graphs from different environments/orchestrator can be combined together.

As an example of a service, let us consider a point-to-point L2VPN connection (i.e., pseudowire). Such a service would take as parameters the two ends of the connection (device, interface or subinterface, and address of the other end) and configure both devices (and maybe more) so that a L2VPN connection is established between the two devices. Examples of symptoms might be "Interface has high error rate" or "Interface flapping", or "Device almost out of memory".

To compute the health status of such a service, the service definition is decomposed into an assurance graph formed by subservices linked through dependencies. Each subservice is then turned into an expression graph that details how to fetch metrics from the devices and compute the health status of the subservice. The subservice expressions are combined according to the dependencies between the subservices in order to obtain the expression graph which computes the health status of the service.

The overall SAIN architecture is presented in Figure 1. Based on the service configuration, the SAIN orchestrator decomposes the assurance graph, to the best of its knowledge. It then sends to the SAIN agents the assurance graph along some other configuration options. The SAIN agents are responsible for building the expression graph and computing the health statuses in a distributed manner. The collector is in charge of collecting and displaying the current inferred health status of the service instances and subservices. Finally, the automation loop is closed by having the SAIN collector providing feedback to the network/service orchestrator.

In order to make agents, orchestrators and collectors from different vendors interoperable, their interface is defined as a YANG model in a companion RFC [I-D.ietf-opsawg-service-assurance-yang]. In Figure 1, the communications that are normalized by this model are tagged with a "Y". The use of these YANG modules is further explained in Section 3.6.

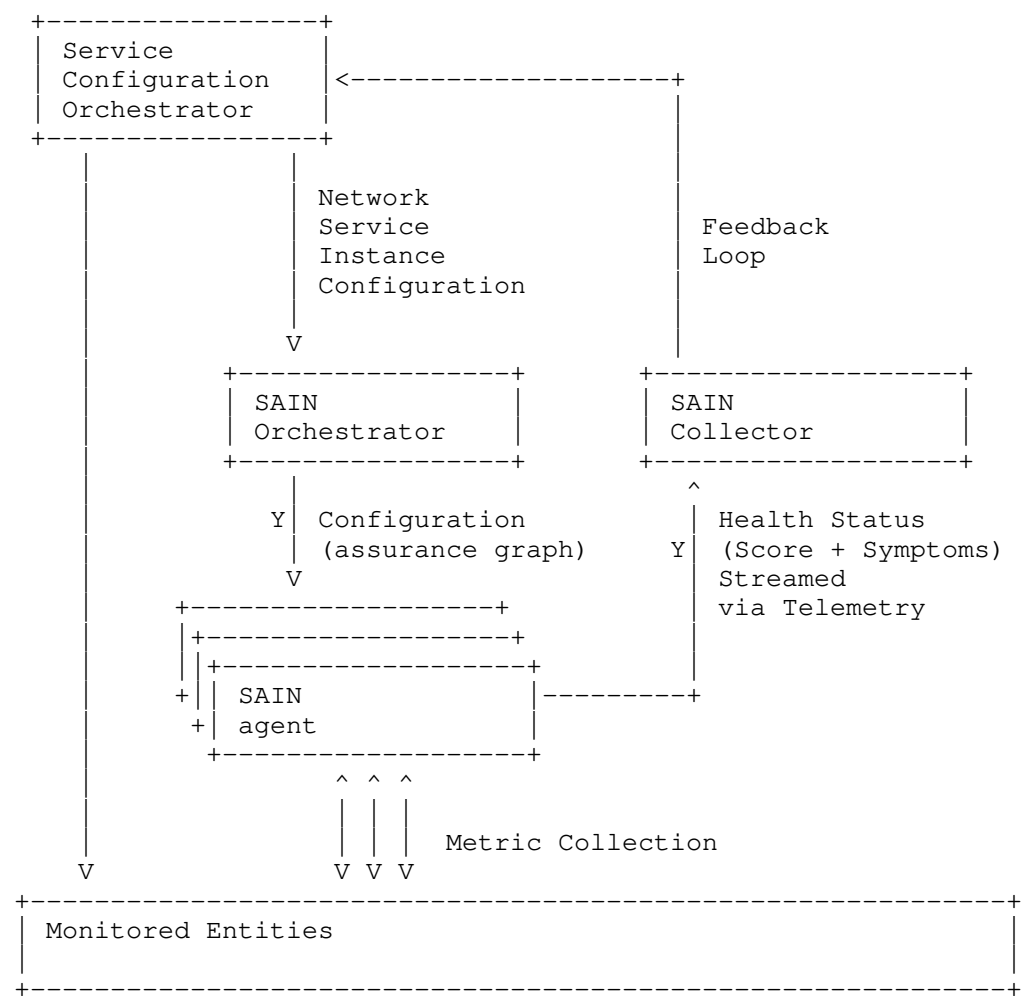


Figure 1: SAIN Architecture

In order to produce the score assigned to a service instance, the architecture performs the following tasks:

- * Analyze the configuration pushed to the network device(s) for configuring the service instance and decide: which information is needed from the device(s), such a piece of information being called a metric, which operations to apply to the metrics for computing the health status.

- * Stream (via telemetry [RFC8641]) operational and config metric values when possible, else continuously poll.
- * Continuously compute the health status of the service instances, based on the metric values.

3.1. Inferring a Service Instance Configuration into an Assurance Graph

In order to structure the assurance of a service instance, the service instance is decomposed into so-called subservice instances. Each subservice instance focuses on a specific feature or subpart of the service.

The decomposition into subservices is an important function of this architecture, for the following reasons.

- * The result of this decomposition provides a relational picture of a service instance, that can be represented as a graph (called assurance graph) to the operator.
- * Subservices provide a scope for particular expertise and thereby enable contribution from external experts. For instance, the subservice dealing with the optics health should be reviewed and extended by an expert in optical interfaces.
- * Subservices that are common to several service instances are reused for reducing the amount of computation needed.

The assurance graph of a service instance is a DAG representing the structure of the assurance case for the service instance. The nodes of this graph are service instances or subservice instances. Each edge of this graph indicates a dependency between the two nodes at its extremities: the service or subservice at the source of the edge depends on the service or subservice at the destination of the edge.

Figure 2 depicts a simplistic example of the assurance graph for a tunnel service. The node at the top is the service instance, the nodes below are its dependencies. In the example, the tunnel service instance depends on the "peer1" and "peer2" tunnel interfaces, which in turn depend on the respective physical interfaces, which finally depend on the respective "peer1" and "peer2" devices. The tunnel service instance also depends on the IP connectivity that depends on the IS-IS routing protocol.

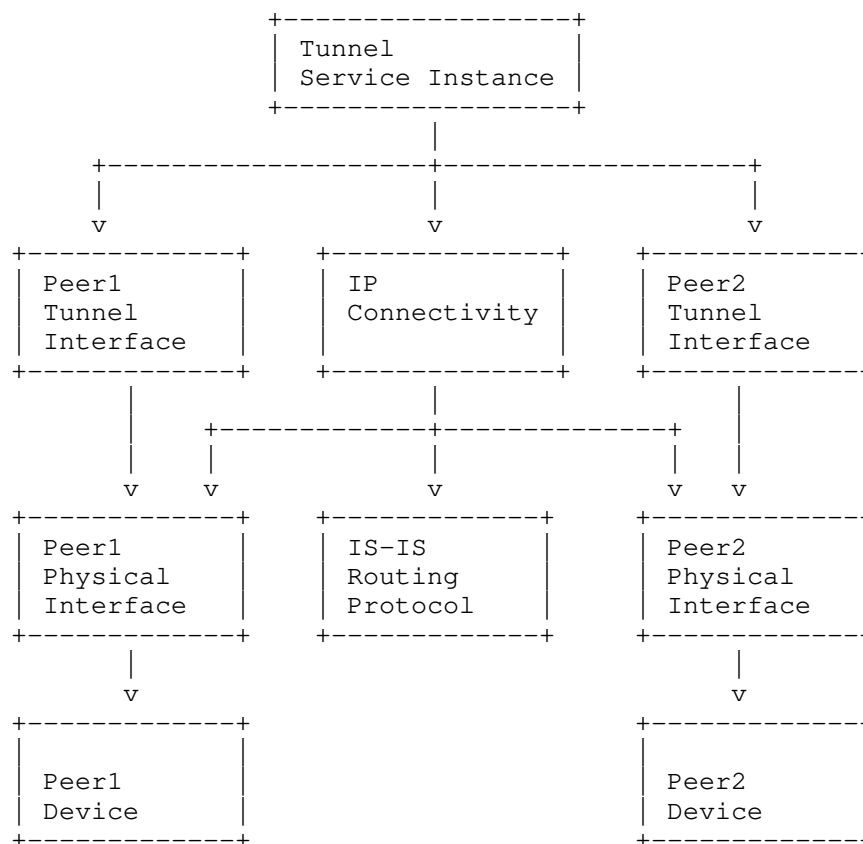


Figure 2: Assurance Graph Example

Depicting the assurance graph helps the operator to understand (and assert) the decomposition. The assurance graph shall be maintained during normal operation with addition, modification and removal of service instances. A change in the network configuration or topology shall be reflected in the assurance graph. As a first example, a change of routing protocol from IS-IS to OSPF would change the assurance graph accordingly. As a second example, assuming that ECMP is in place for the source router for that specific tunnel; in that case, multiple interfaces must now be monitored, on top of the monitoring the ECMP health itself.

3.1.1. Circular Dependencies

The edges of the assurance graph represent dependencies. An assurance graph is a DAG if and only if there are no circular dependencies among the subservices, and every assurance graph should avoid circular dependencies. However, in some cases, circular dependencies might appear in the assurance graph.

First, the assurance graph of a whole system is obtained by combining the assurance graph of every service running on that system. Here combining means that two subservices having the same type and the same parameters are in fact the same subservice and thus a single node in the graph. For instance, the subservice of type "device" with the only parameter (the device id) set to "PE1" will appear only once in the whole assurance graph even if several services rely on that device. Now, if two engineers design assurance graphs for two different services, and engineer A decides that an interface depends on the link it is connected to, but engineer B decides that the link depends on the interface it is connected to, then when combining the two assurance graphs, we will have a circular dependency interface -> link -> interface.

Another case possibly resulting in circular dependencies is when subservices are not properly identified. Assume that we want to assure a kubernetes cluster. If we represent the cluster by a subservice and the network service by another subservice, we will likely model that the network service depends on the cluster, because the network service is orchestrated by kubernetes, and that the cluster depends on the network service because it implements the communications. A finer decomposition might distinguish between the resources for executing containers (a part of our cluster subservice) and the communication between the containers (which could be modelled in the same way as communication between routers).

In any case, it is likely that circular dependencies will show up in the assurance graph. A first step would be to detect circular dependencies as soon as possible in the SAIN architecture. Such a detection could be carried out by the SAIN Orchestrator. Whenever a circular dependency is detected, the newly added service would not be monitored until more careful modelling or alignment between the different teams (engineer A and B) remove the circular dependency.

As more elaborate solution we could consider a graph transformation:

- * Decompose the graph into strongly connected components.
- * For each strongly connected component:

- Remove all edges between nodes of the strongly connected component
- Add a new "top" node for the strongly connected component
- For each edge pointing to a node in the strongly connected component, change the destination to the "top" node
- Add a dependency from the top node to every node in the strongly connected component.

Such an algorithm would include all symptoms detected by any subservice in one of the strongly component and make it available to any subservice that depends on it. Figure 3 shows an example of such a transformation. On the left-hand side, the nodes c, d, e and f form a strongly connected component. The status of a should depend on the status of c, d, e, f, g, and h, but this is hard to compute because of the circular dependency. On the right hand-side, a depends on all this nodes as well, but there the circular dependency has been removed.

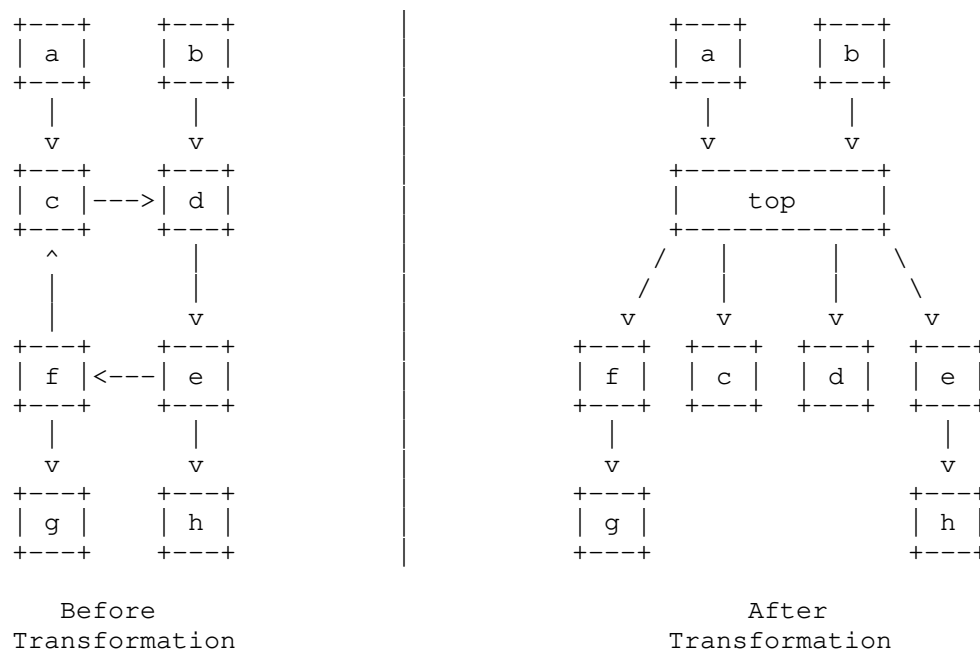


Figure 3: Graph transformation

We consider a concrete example to illustrate this transformation. Let's assume that Engineer A is building an assurance graph dealing with IS-IS and Engineer B is building an assurance graph dealing with OSPF. The graph from Engineer A could contain the following:

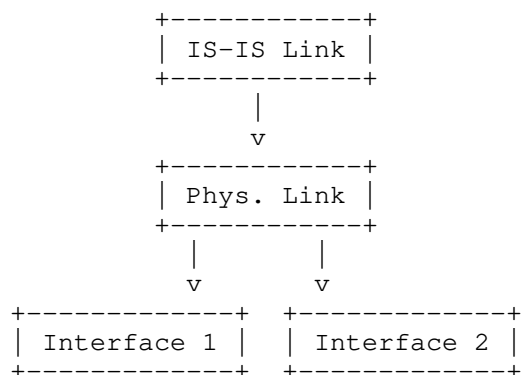


Figure 4: Fragment of assurance graph from Engineer A

The graph from Engineer B could contain the following:

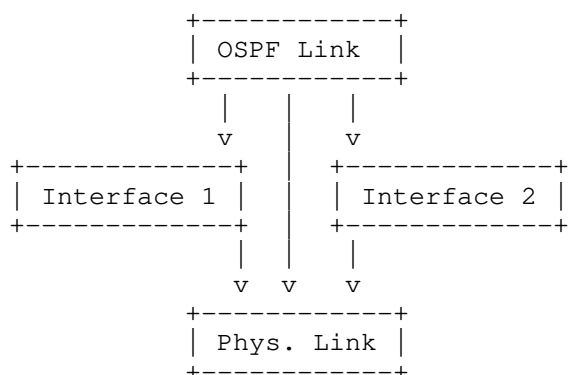


Figure 5: Fragment of assurance graph from Engineer B

Each Interface subservice and the Physical Link subservice are common to both fragments above. Each of these subservice appears only once in the graph merging the two fragments. Dependencies from both fragments are included in the merged graph, resulting in a circular dependency:

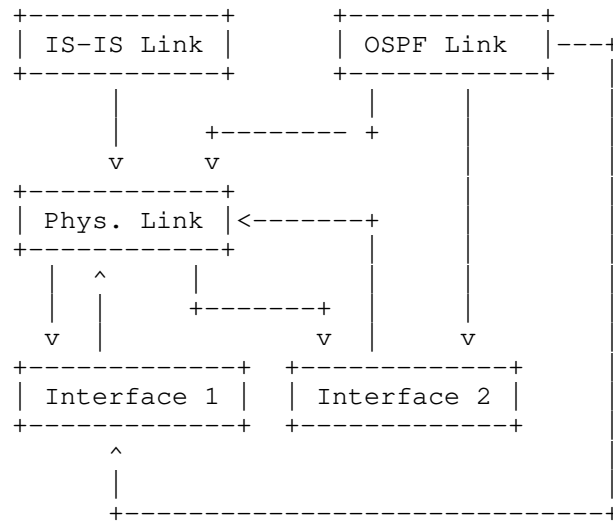


Figure 6: Merging graphs from A and B

The solution presented above would result in graph looking as follows, where a new "empty" node is included. Using that transformation, all dependencies are indirectly satisfied for the nodes outside the circular dependency, in the sense that both IS-IS and OSPF links have indirect dependencies to the two interfaces and the link. However, the dependencies between the link and the interfaces are lost as they were causing the circular dependency.

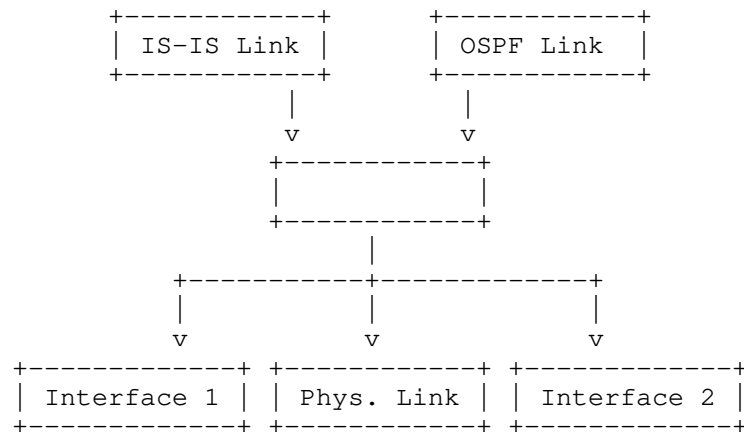


Figure 7: Removing circular dependencies after merging graphs
from A and B

3.2. Intent and Assurance Graph

The SAIN orchestrator analyzes the configuration of a service instance to:

- * Try to capture the intent of the service instance, i.e., what is the service instance trying to achieve.
- * Decompose the service instance into subservices representing the network features on which the service instance relies.

The SAIN orchestrator must be able to analyze configuration from various devices and produce the assurance graph.

To schematize what a SAIN orchestrator does, assume that the configuration for a service instance touches two devices and configure on each device a virtual tunnel interface. Then:

- * Capturing the intent would start by detecting that the service instance is actually a tunnel between the two devices, and stating that this tunnel must be functional. This is the current state of SAIN, however it does not completely capture the intent which might additionally include, for instance, the latency and bandwidth requirements of this tunnel.
- * Decomposing the service instance into subservices would result in the assurance graph depicted in Figure 2, for instance.

In order for SAIN to be applied, the configuration necessary for each service instance should be identifiable and thus should come from a "service-aware" source. While the Figure 1 makes a distinction between the SAIN orchestrator and a different component providing the service instance configuration, in practice those two components are mostly likely combined. The internals of the orchestrator are currently out of scope of this document.

3.3. Subservices

A subservice corresponds to subpart or a feature of the network system that is needed for a service instance to function properly. In the context of SAIN, subservice is actually a shortcut for subservice assurance, that is the method for assuring that a subservice behaves correctly.

Subservices, just as with services, have high-level parameters that specify the type and specific instance to be assured. For example, assuring a device requires the specific `deviceId` as parameter. For example, assuring an interface requires the specific combination of `deviceId` and `interfaceId`.

A subservice is also characterized by a list of metrics to fetch and a list of computations to apply to these metrics in order to infer a health status.

3.4. Building the Expression Graph from the Assurance Graph

From the assurance graph is derived a so-called global computation graph. First, each subservice instance is transformed into a set of subservice expressions that take metrics and constants as input (i.e., sources of the DAG) and produce the status of the subservice, based on some heuristics. Then for each service instance, the service expressions are constructed by combining the subservice expressions of its dependencies. The way service expressions are combined depends on the dependency types (impacting or informational). Finally, the global computation graph is built by combining the service expressions. In other words, the global computation graph encodes all the operations needed to produce health statuses from the collected metrics.

Subservices shall be device independent. To justify this, let's consider the interface operational status. Depending on the device capabilities, this status can be collected by an industry-accepted YANG module (IETF, Openconfig), by a vendor-specific YANG module, or even by a MIB module. If the subservice was dependent on the mechanism to collect the operational status, then we would need multiple subservice definitions in order to support all different mechanisms. This also implies that, while waiting for all the metrics to be available via standard YANG modules, SAIN agents might have to retrieve metric values via non-standard YANG models, via MIB modules, Command Line Interface (CLI), etc., effectively implementing a normalization layer between data models and information models.

In order to keep subservices independent from metric collection method, or, expressed differently, to support multiple combinations of platforms, OSes, and even vendors, the architecture introduces the concept of "metric engine". The metric engine maps each device-independent metric used in the subservices to a list of device-specific metric implementations that precisely define how to fetch values for that metric. The mapping is parameterized by the characteristics (model, OS version, etc.) of the device from which the metrics are fetched.

3.5. Building the Expression from a Subservice

Additionally, to the list of metrics, each subservice defines a list of expressions to apply on the metrics in order to compute the health status of the subservice. The definition or the standardization of those expressions (also known as heuristic) is currently out of scope of this standardization.

3.6. Open Interfaces with YANG Modules

The interfaces between the architecture components are open thanks to the YANG modules specified in YANG Modules for Service Assurance [I-D.ietf-opsawg-service-assurance-yang]; they specify objects for assuring network services based on their decomposition into so-called subservices, according to the SAIN architecture.

This module is intended for the following use cases:

- * Assurance graph configuration:
 - Subservices: configure a set of subservices to assure, by specifying their types and parameters.
 - Dependencies: configure the dependencies between the subservices, along with their types.
- * Assurance telemetry: export the health status of the subservices, along with the observed symptoms.

Some examples of YANG instances can be found in Appendix A of [I-D.ietf-opsawg-service-assurance-yang].

3.7. Handling Maintenance Windows

Whenever network components are under maintenance, the operator want to inhibit the emission of symptoms from those components. A typical use case is device maintenance, during which the device is not supposed to be operational. As such, symptoms related to the device health should be ignored, as well as symptoms related to the device-specific subservices, such as the interfaces, as their state changes is probably the consequence of the maintenance.

To configure network components as "under maintenance" in the SAIN architecture, the ietf-service-assurance model proposed in [I-D.ietf-opsawg-service-assurance-yang] specifies an "under-maintenance" flag per service or subservice instance. When this flag is set and only when this flag is set, the companion field "maintenance-contact" must be set to a string that identifies the

person or process who requested the maintenance. When a service or subservice is flagged as under maintenance, it may report a generic "Under Maintenance" symptom, for propagation towards subservices that depend on this specific subservice: any other symptom from this service, or by one of its impacting dependencies MUST NOT be reported.

We illustrate this mechanism on three independent examples based on the assurance graph depicted in Figure 2:

- * Device maintenance, for instance upgrading the device OS. The operator sets the "under-maintenance" flag for the subservice "Peer1" device. This inhibits the emission of symptoms from "Peer1 Physical Interface", "Peer1 Tunnel Interface" and "Tunnel Service Instance". All other subservices are unaffected.
- * Interface maintenance, for instance replacing a broken optic. The operator sets the "under-maintenance" flag for the subservice "Peer1 Physical Interface". This inhibits the emission of symptoms from "Peer 1 Tunnel Interface" and "Tunnel Service Instance". All other subservices are unaffected.
- * Routing protocol maintenance, for instance modifying parameters or redistribution. The operator sets the "under-maintenance" flag for the subservice "IS-IS Routing Protocol". This inhibits the emission of symptoms from "IP connectivity" and "Tunnel Service Instance". All other subservices are unaffected.

3.8. Flexible Architecture

The SAIN architecture is flexible in terms of components. While the SAIN architecture in Figure 1 makes a distinction between two components, the SAIN configuration orchestrator and the SAIN orchestrator, in practice those two components are mostly likely combined. Similarly, the SAIN agents are displayed in Figure 1 as being separate components. Practically, the SAIN agents could be either independent components or directly integrated in monitored entities. A practical example is an agent in a router.

The SAIN architecture is also flexible in terms of services and subservices. Most examples in this document deal with the notion of Network Service YANG modules, with well-known service such as L2VPN or tunnels. However, the concepts of services is general enough to cross into different domains. One of them is the domain of service management on network elements, with also requires its own assurance. Examples includes a DHCP server on a Linux server, a data plane, an IPFIX export, etc. The notion of "service" is generic in this architecture. Indeed, a configured service can itself be a

subservice for someone else. Exactly like a DHCP server/ data plane/ IPFIX export can be considered as subservices for a device, exactly like an routing instance can be considered as a subservice for a L3VPN, exactly like a tunnel can considered as a subservice for an application in the cloud. Exactly like a service function can be be considered as a subservice for a service function chain [RFC7665]. The assurance graph is created to be flexible and open, regardless of the subservice types, locations, or domains.

The SAIN architecture is also flexible in terms of distributed graphs. As shown in Figure 1, our architecture comprises several agents. Each agent is responsible for handling a subgraph of the assurance graph. The collector is responsible for fetching the subgraphs from the different agents and gluing them together. As an example, in the graph from Figure 2, the subservices relative to Peer 1 might be handled by a different agent than the subservices relative to Peer 2 and the Connectivity and IS-IS subservices might be handled by yet another agent. The agents will export their partial graph and the collector will stitch them together as dependencies of the service instance.

And finally, the SAIN architecture is flexible in terms of what it monitors. Most, if not all examples, in this document refer to physical components but this is not a constrain. Indeed, the assurance of virtual components would follow the same principles and an assurance graph composed of virtualized components (or a mix of virtualized and physical ones) is well possible within this architecture.

3.9. Timing

The SAIN architecture requires time synchronization, with Network Time Protocol (NTP) [RFC5905] as a candidate, between all elements: monitored entities, SAIN agents, Service Configuration Orchestrator, the SAIN collector, as well as the SAIN Orchestrator. This guarantees the correlations of all symptoms in the system, correlated with the right assurance graph version.

The SAIN agent might have to remove some symptoms for specific subservice symptoms, because there are outdated and not relevant any longer, or simply because the SAIN agent needs to free up some space. Regardless of the reason, it's important for a SAIN collector (re-)connecting to a SAIN agent to understand the effect of this garbage collection. Therefore, the SAIN agent contains a YANG object specifying the date and time at which the symptoms history starts for the subservice instances.

3.10. New Assurance Graph Generation

The assurance graph will change along the time, because services and subservices come and go (changing the dependencies between subservices), or simply because a subservice is now under maintenance. Therefore an assurance graph version must be maintained, along with the date and time of its last generation. The date and time of a particular subservice instance (again dependencies or under maintenance) might be kept. From a client point of view, an assurance graph change is triggered by the value of the assurance-graph-version and assurance-graph-last-change YANG leaves. At that point in time, the client (collector) follows the following process:

- * Keep the previous assurance-graph-last-change value (let's call it time T)
- * Run through all subservice instance and process the subservice instances for which the last-change is newer than the time T
- * Keep the new assurance-graph-last-change as the new referenced date and time

4. Security Considerations

The SAIN architecture helps operators to reduce the mean time to detect and mean time to repair. As such, it should not cause any security threats. However, the SAIN agents must be secure: a compromised SAIN agent could be sending wrong root causes or symptoms to the management systems.

Except for the configuration of telemetry, the agents do not need "write access" to the devices they monitor. This configuration is applied with a YANG module, whose protection is covered by Secure Shell (SSH) [RFC6242] for NETCONF or TLS [RFC8446] for RESTCONF.

The data collected by SAIN could potentially be compromising to the network or provide more insight into how the network is designed. Considering the data that SAIN requires (including CLI access in some cases), one should weigh data access concerns with the impact that reduced visibility will have on being able to rapidly identify root causes.

If a closed loop system relies on this architecture then the well known issue of those system also applies, i.e., a lying device or compromised agent could trigger partial reconfiguration of the service or network. The SAIN architecture neither augments or reduces this risk.

5. IANA Considerations

This document includes no request to IANA.

6. Contributors

* Youssef El Fathi

* Eric Vyncke

7. Open Issues

Refer to the Intent-based Networking NMRG documents (Intent Assurance, Service Intent: synonym for custom service model see [I-D.irtf-nmrg-ibn-concepts-definitions] and [I-D.irtf-nmrg-ibn-intent-classification]).

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>>.
- [RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <<https://www.rfc-editor.org/info/rfc5905>>.
- [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>>.

8.2. Informative References

- [I-D.ietf-opsawg-service-assurance-yang] Claise, B., Quilbeuf, J., Lucente, P., Fasano, P., and T. Arumugam, "YANG Modules for Service Assurance", Work in Progress, Internet-Draft, draft-ietf-opsawg-service-assurance-yang-02, 4 January 2022, <<https://www.ietf.org/archive/id/draft-ietf-opsawg-service-assurance-yang-02.txt>>.
- [I-D.irtf-nmrg-ibn-concepts-definitions] Clemm, A., Ciavaglia, L., Granville, L. Z., and J. Tantsura, "Intent-Based Networking - Concepts and

Definitions", Work in Progress, Internet-Draft, draft-irtf-nmrg-ibn-concepts-definitions-06, 15 December 2021, <<https://www.ietf.org/archive/id/draft-irtf-nmrg-ibn-concepts-definitions-06.txt>>.

[I-D.irtf-nmrg-ibn-intent-classification]

Li, C., Havel, O., Olariu, A., Martinez-Julia, P., Nobre, J. C., and D. R. Lopez, "Intent Classification", Work in Progress, Internet-Draft, draft-irtf-nmrg-ibn-intent-classification-06, 22 February 2022, <<https://www.ietf.org/archive/id/draft-irtf-nmrg-ibn-intent-classification-06.txt>>.

[Piovesan2017]

Piovesan, A. and E. Griffor, "Reasoning About Safety and Security: The Logic of Assurance", 2017.

[RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, DOI 10.17487/RFC2865, June 2000, <<https://www.rfc-editor.org/info/rfc2865>>.

[RFC3164] Lonvick, C., "The BSD Syslog Protocol", RFC 3164, DOI 10.17487/RFC3164, August 2001, <<https://www.rfc-editor.org/info/rfc3164>>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.

[RFC7011] Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, DOI 10.17487/RFC7011, September 2013, <<https://www.rfc-editor.org/info/rfc7011>>.

[RFC7149] Boucadair, M. and C. Jacquenet, "Software-Defined Networking: A Perspective from within a Service Provider Environment", RFC 7149, DOI 10.17487/RFC7149, March 2014, <<https://www.rfc-editor.org/info/rfc7149>>.

- [RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <<https://www.rfc-editor.org/info/rfc7665>>.
- [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>>.
- [RFC8199] Bogdanovic, D., Claise, B., and C. Moberg, "YANG Module Classification", RFC 8199, DOI 10.17487/RFC8199, July 2017, <<https://www.rfc-editor.org/info/rfc8199>>.
- [RFC8309] Wu, Q., Liu, W., and A. Farrel, "Service Models Explained", RFC 8309, DOI 10.17487/RFC8309, January 2018, <<https://www.rfc-editor.org/info/rfc8309>>.
- [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>>.
- [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>>.
- [RFC8907] Dahm, T., Ota, A., Medway Gash, D.C., Carrel, D., and L. Grant, "The Terminal Access Controller Access-Control System Plus (TACACS+) Protocol", RFC 8907, DOI 10.17487/RFC8907, September 2020, <<https://www.rfc-editor.org/info/rfc8907>>.
- [RFC8969] Wu, Q., Ed., Boucadair, M., Ed., Lopez, D., Xie, C., and L. Geng, "A Framework for Automating Service and Network Management with YANG", RFC 8969, DOI 10.17487/RFC8969, January 2021, <<https://www.rfc-editor.org/info/rfc8969>>.

Appendix A. Changes between revisions

v00 - v01

- * Cover the feedback received during the WG call for adoption

Acknowledgements

The authors would like to thank Stephane Litkowski, Charles Eckel, Rob Wilton, Vladimir Vassiliev, Gustavo Alburquerque, Stefan Vallin, Eric Vyncke, and Mohamed Boucadair for their reviews and feedback.

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

Dan Voyer
Bell Canada
Canada
Email: daniel.voyer@bell.ca

Thangam Arumugam
Cisco Systems, Inc.
Milpitas (California),
United States of America
Email: tarumuga@cisco.com

OPSAWG
Internet-Draft
Intended status: Standards Track
Expires: 31 October 2022

B. Claise
J. Quilbeuf
Huawei
P. Lucente
NTT
P. Fasano
TIM S.p.A
T. Arumugam
Cisco Systems, Inc.
29 April 2022

YANG Modules for Service Assurance
draft-ietf-opsawg-service-assurance-yang-05

Abstract

This document specifies YANG modules representing assurance graphs. These graphs represent the assurance of a given service by decomposing it into atomic assurance elements called subservices. A companion RFC, Service Assurance for Intent-based Networking Architecture, presents an architecture for implementing the assurance of such services.

The YANG data models in this document conforms to the Network Management Datastore Architecture (NMDA) defined in RFC 8342.

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 31 October 2022.

Copyright Notice

Copyright (c) 2022 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	3
1.1. Terminology	3
2. YANG Models Overview	3
3. Base ietf-service-assurance YANG module	4
3.1. Tree View	4
3.2. Concepts	5
3.3. YANG Module	7
4. Subservice Extension: ietf-service-assurance-device YANG module	15
4.1. Tree View	15
4.2. Complete Tree View	15
4.3. Concepts	16
4.4. YANG Module	17
5. Subservice Extension: ietf-service-assurance-interface YANG module	19
5.1. Tree View	19
5.2. Complete Tree View	19
5.3. Concepts	20
5.4. YANG Module	21
6. Security Considerations	23
7. IANA Considerations	23
7.1. The IETF XML Registry	24
7.2. The YANG Module Names Registry	24
8. Open Issues	24
9. References	24
9.1. Normative References	24
9.2. Informative References	25
Appendix A. Vendor-specific Subservice Extension: example-service-assurance-device-acme YANG module	26
A.1. Tree View	26
A.2. Complete Tree View	26
A.3. Concepts	28
A.4. YANG Module	28
Appendix B. Further Extensions: IP Connectivity and IS-IS subservices	30
B.1. IP Connectivity Tree View	30
B.2. IS-IS Tree View	31

B.3. Global Tree View	31
B.4. IP Connectivity YANG Module	32
B.5. IS-IS YANG Module	35
Appendix C. Example of YANG instances	37
Appendix D. YANG Library for Service Assurance	40
Appendix E. Changes between revisions	42
Acknowledgements	43
Authors' Addresses	43

1. Introduction

The "Service Assurance for Intent-based Networking Architecture" [I-D.ietf-opsawg-service-assurance-architecture], specifies the architecture and all of its components for service assurance. This document complements the architecture by providing open interfaces between components. More specifically, the goal is to provide YANG modules for the purpose of service assurance in a format that is:

- * machine readable
- * vendor independent
- * augmentable

1.1. 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 13 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The terms used in this document are defined in [I-D.ietf-opsawg-service-assurance-architecture]

2. YANG Models Overview

The main YANG module, `ietf-service-assurance`, defines objects for assuring network services based on their decomposition into so-called subservices. The subservices are hierarchically organised by dependencies. The subservices, along with the dependencies, constitute an assurance graph. This module should be supported by an agent, able to interact with the devices in order to produce a health status and symptoms for each subservice in the assurance graph. This module is intended for the following use cases:

- * Assurance graph configuration:

- Subservices: configure a set of subservices to assure, by specifying their types and parameters.
 - Dependencies: configure the dependencies between the subservices, along with their type.
- * Assurance telemetry: export the health status of the subservices, along with the observed symptoms.

The main module represents the configuration (subservice and dependencies) and operational data (health status and symptoms) in a single tree. Other modules follow the same pattern. Thus, the modules presented in this document conform to the Network Management Datastore Architecture defined in [RFC8342].

The second YANG module, `ietf-service-assurance-device`, extends the `ietf-service-assurance` module to add support for the device subservice. Additional subservice types might be added the same way.

The third YANG module, `ietf-service-assurance-interface`, is another example that extends the `ietf-service-assurance` module. This extension adds support for the interface subservice.

We provide additional examples in the appendix. The module `example-service-assurance-device-acme` extends the `ietf-service-assurance-device` module to customize it for devices of the fictional ACME Corporation. Additional vendor-specific parameters might be added the same way. We also provide the modules `example-service-assurance-ip-connectivity` and `example-service-assurance-is-is` to completely model the example from the SAIN architecture draft [I-D.ietf-opsawg-service-assurance-architecture].

3. Base `ietf-service-assurance` YANG module

3.1. Tree View

The following tree diagram [RFC8340] provides an overview of the `ietf-service-assurance` data model.

```

module: ietf-service-assurance
  +--ro assurance-graph-version          yang:counter64
  +--ro assurance-graph-last-change      yang:date-and-time
  +--rw subservices
    +--rw subservice* [type id]
      +--rw type                          identityref
      +--rw id                            string
      +--ro last-change?                  yang:date-and-time
      +--ro label?                        string
      +--rw under-maintenance?            boolean
      +--rw maintenance-contact           string
      +--rw (parameter)?
        +--:(service-instance-parameter)
          +--rw service-instance-parameter
            +--rw service                  string
            +--rw instance-name            string
      +--ro health-score?                 union
      +--ro symptoms-history-start?       yang:date-and-time
      +--rw symptoms
        +--ro symptom* [start-date-time id]
          +--ro id                        string
          +--ro health-score-weight?      uint8
          +--ro description?              string
          +--ro start-date-time           yang:date-and-time
          +--ro stop-date-time?           yang:date-and-time
      +--rw dependencies
        +--rw dependency* [type id]
          +--rw type
            | -> /subservices/subservice/type
          +--rw id                        leafref
          +--rw dependency-type?          identityref

```

3.2. Concepts

The ietf-service-assurance YANG model assumes an identified number of subservices, to be assured independently. A subservice is a feature or a subpart of the network system that a given service instance might depend on. Example of subservices include:

- * **device:** whether a device is healthy, and if not, what are the symptoms. Potential symptoms are "CPU overloaded", "Out of RAM", or "Out of TCAM".
- * **ip-connectivity:** given two IP addresses owned by two devices, what is the quality of the connection between them. Potential symptoms are "No route available" or "ECMP Imbalance".

The first example is a subservice representing a subpart of the network system, while the second is a subservice representing a feature of the network. In both cases, these subservices might depend on other subservices, for instance, the connectivity might depend on a subservice representing the routing mechanism and on a subservice representing ECMP.

The status of each subservice contains a list of symptoms. Each symptom is specified by a unique id and contains a health-score-weight (the impact to the health score incurred by this symptom), a label (text describing what the symptom is), and dates and times at which the symptom was detected and stopped being detected. While the unique id is sufficient as an unique key list, the start-date-time second key help sorting and retrieving relevant symptoms.

The relation between the health score and the health-score-weight of the currently active symptoms is not explicitly defined in this draft. The only requirement is that a non-maximal score must be explained by at least one symptom. A way to enforce that requirement is to first detect symptoms and then compute the health score based on the health-score-weight of the detected symptoms. As an example, this computation could be to sum the health-score-weight of the active symptoms, subtract that value from 100 and change the value to 0 if negative. The relation between health-score and health-score-weight is left to the implementor (of an agent [I-D.ietf-opsawg-service-assurance-architecture]). To consider for implementing this relation: the health-score is mostly for humans, the symptoms are what the closed loop automation can build on.

The assurance of a given service instance can be obtained by composing the assurance of the subservices that it depends on, via the dependency relations.

A subservice declaration MUST provide:

- * A type: identity inheriting of the base identity for subservice,
- * An id: string uniquely identifying the subservice among those with the same identity,
- * One or more parameters, which should be specified in an augmenting model, as described in the next sections.

The type and id uniquely identify a given subservice. They are used to indicate the dependencies. Dependencies have types as well. Two types are specified in the model:

- * **Impacting:** such a dependency indicates an impact on the health of the dependent,
- * **Informational:** such a dependency might explain why the dependent has issues but does not impact its health.

To illustrate the difference between "impacting" and "informational", consider the interface subservice, representing a network interface. If the device to which the network interface belongs goes down, the network interface will transition to a down state as well. Therefore, the dependency of the interface subservice towards the device subservice is "impacting". On the other hand, a dependency towards the ecmp-load subservice, which checks that the load between ECMP remains stable throughout time, is only "informational". Indeed, services might be perfectly healthy even if the load distribution between ECMP changed. However, such an instability might be a relevant symptom for diagnosing the root cause of a problem.

Service instances **MUST** be modeled as a particular type of subservice with two parameters, a type and an instance name. The type is the name of the service defined in the network orchestrator, for instance "point-to-point-l2vpn". The instance name is the name assigned to the particular instance to be assured, for instance the name of the customer using that instance.

The "under-maintenance" and "maintenance-contact" flags inhibit the emission of symptoms for that subservice and subservices that depend on them. See Section 3.7 of [I-D.ietf-opsawg-service-assurance-architecture] for a more detailed discussion.

By specifying service instances and their dependencies in terms of subservices, one defines the whole assurance to apply for them. An assurance agent supporting this model should then produce telemetry in return with, for each subservice: a health-status indicating how healthy the subservice is and when the subservice is not healthy, a list of symptoms explaining why the subservice is not healthy.

3.3. YANG Module

```
<CODE BEGINS> file "ietf-service-assurance@2022-04-07.yang"
```



```
module ietf-service-assurance {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-service-assurance";
  prefix sain;

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

  organization
    "IETF OPSAWG 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.quilbeu@huawei.com>";
  description
    "This module defines objects for assuring network services based on
    their decomposition into so-called subservices, according to the
    SAIN (Service Assurance for Intent-based Networking) architecture.

    The subservices hierarchically organised by dependencies constitute
    an assurance graph. This module should be supported by an assurance
    agent, able to interact with the devices in order to produce a
    health status and symptoms for each subservice in the assurance
    graph.

    This module is intended for the following use cases:
    * Assurance graph configuration:
      - subservices: configure a set of subservices to assure, by
        specifying their types and parameters.
      - dependencies: configure the dependencies between the
        subservices, along with their type.
    * Assurance telemetry: export the health status of the subservices,
      along with the observed symptoms.

    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 2022-04-07 {
  description
    "Shorten prefix. Fix copyright.
     Fix module description";
  reference
    "RFC xxxx: YANG Modules for Service Assurance";
}
revision 2022-01-04 {
  description
    "Explicitely model a missing value";
  reference
    "RFC xxxx: YANG Modules for Service Assurance";
}
revision 2021-06-28 {
  description
    "Made service-instance parameters mandatory.";
  reference
    "RFC xxxx: YANG Modules for Service Assurance";
}
revision 2020-01-13 {
  description
    "Added the maintenance window concept.";
  reference
    "RFC xxxx: YANG Modules for Service Assurance";
}
revision 2019-11-16 {
  description
    "Initial revision.";
  reference
    "RFC xxxx: YANG Modules for Service Assurance";
}

identity subservice-idty {
  description
    "Root identity for all subservice types.";
}

identity service-instance-idty {
  base subservice-idty;
```

```
    description
      "Identity representing a service instance.";
  }

  identity dependency-type {
    description
      "Base identity for representing dependency types.";
  }

  identity informational-dependency {
    base dependency-type;
    description
      "Indicates that symptoms of the dependency might be of interest
       for the dependent, but the status of the dependency should not
       have any impact on the dependent.";
  }

  identity impacting-dependency {
    base dependency-type;
    description
      "Indicates that the status of the dependency directly impacts the
       status of the dependent.";
  }

  grouping symptom {
    description
      "Contains the list of symptoms for a specific subservice.";
    leaf id {
      type string;
      description
        "A unique identifier for the symptom.";
    }
    leaf health-score-weight {
      type uint8 {
        range "0 .. 100";
      }
      description
        "The weight to the health score incurred by this symptom. The
         higher the value, the more of an impact this symptom has. If a
         subservice health score is not 100, there must be at least one
         symptom with a health score weight larger than 0.";
    }
    leaf description {
      type string;
      description
        "Description of the symptom, i.e. text describing what the
         symptom is, to be computer-consumable and be displayed on a
         human interface. ";
    }
  }
```

```
    }
    leaf start-date-time {
        type yang:date-and-time;
        description
            "Date and time at which the symptom was detected.";
    }
    leaf stop-date-time {
        type yang:date-and-time;
        description
            "Date and time at which the symptom stopped being detected.";
    }
}

grouping subservice-dependency {
    description
        "Represent a dependency to another subservice.";
    leaf type {
        type leafref {
            path "/subservices/subservice/type";
        }
        description
            "The type of the subservice to refer to (e.g. device).";
    }
    leaf id {
        type leafref {
            path "/subservices/subservice[type=current()/../type]/id";
        }
        description
            "The identifier of the subservice to refer to.";
    }
    leaf dependency-type {
        type identityref {
            base dependency-type;
        }
        description
            "Represents the type of dependency (i.e. informational,
            impacting).";
    }
    // Augment here to add parameters specific to a new dependency-type.
    // For instance, a specific dependency type could keep symptom
    // whose health-score-weight is larger than a given value.
}

leaf assurance-graph-version {
    type yang:counter64;
    config false;
    mandatory true;
    description
```

```
    "The assurance graph version, which increases by 1 for each new
    version, after the changes (dependencies and/or maintenance
    windows parameters) are applied to the subservice(s).";
}
leaf assurance-graph-last-change {
  type yang:date-and-time;
  config false;
  mandatory true;
  description
    "Date and time at which the assurance graph last changed after the
    changes (dependencies and/or maintenance windows parameters) are
    applied to the subservice(s). These date and time must be more
    recent or equal compared to the more recent value of any changed
    subservices last-change";
}
container subservices {
  description
    "Root container for the subservices.";
  list subservice {
    key "type id";
    description
      "List of subservice configured.";
    leaf type {
      type identityref {
        base subservice-idty;
      }
      description
        "Type of the subservice, for instance device or interface.";
    }
    leaf id {
      type string;
      description
        "Unique identifier of the subservice instance, for each
        type.";
    }
    leaf last-change {
      type yang:date-and-time;
      config false;
      description
        "Date and time at which the assurance graph for this
        subservice instance last changed, i.e. dependencies and/or
        maintenance windows parameters.";
    }
    leaf label {
      type string;
      config false;
      description
        "Label of the subservice, i.e. text describing what the
```

```
        subservice is to be displayed on a human interface.";
    }
    leaf under-maintenance {
        type boolean;
        default "false";
        description
            "An optional flag indicating whether this particular
            subservice is under maintenance. Under this circumstance, the
            subservice symptoms and the symptoms of its dependencies in
            the assurance graph should not be taken into account.
            Instead, the subservice should send a 'Under Maintenance'
            single symptom.

            The operator changing the under-maintenance value must set
            the maintenance-contact variable.

            When the subservice is not under maintenance any longer, the
            under-maintenance flag must return to its default value and
            the under-maintenance-owner variable deleted.";
    }
    leaf maintenance-contact {
        when "../under-maintenance = 'true'";
        type string;
        mandatory true;
        description
            "A string used to model an administratively assigned name of
            the resource that changed the under-maintenance value to
            'true'.

            It is suggested that this name contain one or more of the
            following: IP address, management station name,
            network manager's name, location, or phone number. In some
            cases the agent itself will be the owner of an entry. In
            these cases, this string shall be set to a string starting
            with 'monitor'.";
    }
    choice parameter {
        description
            "Specify the required parameters per subservice type.";
        container service-instance-parameter {
            when "derived-from-or-self(..../type,
                'sain:service-instance-idty')";
            description
                "Specify the parameters of a service instance.";
            leaf service {
                type string;
                mandatory true;
                description
```

```
        "Name of the service.";
    }
    leaf instance-name {
        type string;
        mandatory true;
        description
            "Name of the instance for that service.";
    }
}
// Other modules can augment their own cases into here
}
leaf health-score {
    type union {
        type uint8 {
            range "0 .. 100";
        }
        type enumeration {
            enum missing {
                value -1;
                description
                    "Explicitly represent the fact that the health score is
                    missing. This could be used when metrics crucial to
                    establish the health score are not collected anymore.";
            }
        }
    }
}
config false;
description
    "Score value of the subservice health. A value of 100 means
    that subservice is healthy. A value of 0 means that the
    subservice is broken. A value between 0 and 100 means that
    the subservice is degraded.";
}
leaf symptoms-history-start {
    type yang:date-and-time;
    config false;
    description
        "Date and time at which the symptoms history starts for this
        subservice instance, either because the subservice instance
        started at that date and time or because the symptoms before
        that were removed due to a garbage collection process.";
}
container symptoms {
    description
        "Symptoms for the subservice.";
    list symptom {
        key "start-date-time id";
        config false;
    }
}
```

```

        description
            "List of symptoms the subservice. While the start-date-time
            key is not necessary per se, this would get the entries
            sorted by start-date-time for easy consumption.";
        uses symptom;
    }
}
container dependencies {
    description
        "configure the dependencies between the subservices, along
        with their types.";
    list dependency {
        key "type id";
        description
            "List of soft dependencies of the subservice.";
        uses subservice-dependency;
    }
}
}
}
}

```

<CODE ENDS>

4. Subservice Extension: ietf-service-assurance-device YANG module

4.1. Tree View

The following tree diagram [RFC8340] provides an overview of the ietf-service-assurance-device data model.

module: ietf-service-assurance-device

```

augment /sain:subservices/sain:subservice/sain:parameter:
  +--rw parameters
    +--rw device      string

```

4.2. Complete Tree View

The following tree diagram [RFC8340] provides an overview of the ietf-service-assurance and ietf-service-assurance-device data models.


```

module: ietf-service-assurance
  +--ro assurance-graph-version          yang:counter64
  +--ro assurance-graph-last-change      yang:date-and-time
  +--rw subservices
    +--rw subservice* [type id]
      +--rw type                          identityref
      +--rw id                            string
      +--ro last-change?                  yang:date-and-time
      +--ro label?                        string
      +--rw under-maintenance?            boolean
      +--rw maintenance-contact           string
      +--rw (parameter)?
        +--:(service-instance-parameter)
          +--rw service-instance-parameter
            +--rw service                  string
            +--rw instance-name            string
        +--:(sain-device:parameters)
          +--rw sain-device:parameters
            +--rw sain-device:device      string
      +--ro health-score?                  union
      +--ro symptoms-history-start?        yang:date-and-time
      +--rw symptoms
        +--ro symptom* [start-date-time id]
          +--ro id                        string
          +--ro health-score-weight?      uint8
          +--ro description?              string
          +--ro start-date-time            yang:date-and-time
          +--ro stop-date-time?            yang:date-and-time
      +--rw dependencies
        +--rw dependency* [type id]
          +--rw type
            | -> /subservices/subservice/type
          +--rw id                        leafref
          +--rw dependency-type?          identityref

```

4.3. Concepts

As the number of subservices will grow over time, the YANG module is designed to be extensible. A new subservice type requires the precise specifications of its type and expected parameters. Let us illustrate the example of the new device subservice type. As the name implies, it monitors and reports the device health, along with some symptoms in case of degradation.

For our device subservice definition, the new identity device-idty is specified, as an inheritance from the base identity for subservices. This indicates to the assurance agent that we are now assuring the health of a device.

The typical parameter for the configuration of the device subservice is the name of the device that we want to assure. By augmenting the parameter choice from ietf-service-assurance YANG module for the case of the device-idty subservice type, this new parameter is specified.

4.4. YANG Module

<CODE BEGINS> file "ietf-service-assurance-device@2022-04-07.yang"

```
module ietf-service-assurance-device {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-service-assurance-device";
  prefix sain-device;

  import ietf-service-assurance {
    prefix sain;
    reference
      "RFC xxxx: YANG Modules for Service Assurance";
  }

  organization
    "IETF OPSAWG 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 extends the ietf-service-assurance module to add
    support for the device subservice.
```

Checks whether a network device is healthy.

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 2022-04-07 {
  description
    "Fix mandatory in augment error by moving when clause.
    Shorten prefix. Fix module description.
    Fix module description";
  reference
    "RFC xxxx: YANG Modules for Service Assurance";
}
revision 2021-06-28 {
  description
    "Renamed the container for parameters.";
  reference
    "RFC xxxx: YANG Modules for Service Assurance";
}
revision 2020-01-13 {
  description
    "Added the maintenance window concept.";
  reference
    "RFC xxxx: YANG Modules for Service Assurance";
}
revision 2019-11-16 {
  description
    "Initial revision.";
  reference
    "RFC xxxx: YANG Modules for Service Assurance";
}

identity device-idty {
  base sain:subservice-idty;
  description
    "Network Device is healthy.";
}

augment "/sain:subservices/sain:subservice/sain:parameter" {
  when "derived-from-or-self(sain:type, 'device-idty')";
  description
    "Specify the required parameters for a new subservice type";
  container parameters {
    description
```

```
        "Specify the required parameters for the device-idty
        subservice type";
    leaf device {
        type string;
        mandatory true;
        description
            "The device to monitor.";
    }
}
}
```

<CODE ENDS>

5. Subservice Extension: ietf-service-assurance-interface YANG module

5.1. Tree View

The following tree diagram [RFC8340] provides an overview of the ietf-service-assurance-interface data model.

```
module: ietf-service-assurance-interface

  augment /sain:subservices/sain:subservice/sain:parameter:
    +--rw parameters
      +--rw device      string
      +--rw interface   string
```

5.2. Complete Tree View

The following tree diagram [RFC8340] provides an overview of the ietf-service-assurance, ietf-service-assurance-device, and ietf-service-assurance-interface data models.

```

module: ietf-service-assurance
+--ro assurance-graph-version          yang:counter64
+--ro assurance-graph-last-change      yang:date-and-time
+--rw subservices
  +--rw subservice* [type id]
    +--rw type                          identityref
    +--rw id                            string
    +--ro last-change?                  yang:date-and-time
    +--ro label?                        string
    +--rw under-maintenance?            boolean
    +--rw maintenance-contact           string
    +--rw (parameter)?
      +--:(service-instance-parameter)
        +--rw service-instance-parameter
          +--rw service                  string
          +--rw instance-name            string
      +--:(sain-interface:parameters)
        +--rw sain-interface:parameters
          +--rw sain-interface:device    string
          +--rw sain-interface:interface string
      +--:(sain-device:parameters)
        +--rw sain-device:parameters
          +--rw sain-device:device        string
    +--ro health-score?                 union
    +--ro symptoms-history-start?        yang:date-and-time
+--rw symptoms
  +--ro symptom* [start-date-time id]
    +--ro id                            string
    +--ro health-score-weight?          uint8
    +--ro description?                  string
    +--ro start-date-time                yang:date-and-time
    +--ro stop-date-time?                yang:date-and-time
+--rw dependencies
  +--rw dependency* [type id]
    +--rw type
      | -> /subservices/subservice/type
    +--rw id                            leafref
    +--rw dependency-type?              identityref

```

5.3. Concepts

For our interface subservice definition, the new interface-idty is specified, as an inheritance from the base identity for subservices. This indicates to the assurance agent that we are now assuring the health of an interface.

The typical parameters for the configuration of the interface subservice are the name of the device and, on that specific device, a specific interface. By augmenting the parameter choice from ietf-service-assurance YANG module for the case of the interface-idty subservice type, those two new parameters are specified.

5.4. YANG Module

<CODE BEGINS> file "ietf-service-assurance-interface@2022-04-07.yang"

```
module ietf-service-assurance-interface {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-service-assurance-interface";
  prefix sain-interface;

  import ietf-service-assurance {
    prefix sain;
    reference
      "RFC xxxx: YANG Modules for Service Assurance";
  }

  organization
    "IETF OPSAWG 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 extends the ietf-service-assurance module to add
    support for the interface subservice.
```

Checks whether an interface is healthy.

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 2022-04-07 {
  description
    "Fix mandatory in augment error by moving when clause.
    Shorten prefix. Fix module description.
    Fix module description";
  reference
    "RFC xxxx: YANG Modules for Service Assurance";
}
revision 2021-06-28 {
  description
    "Regroup parameters in a container.";
  reference
    "RFC xxxx: YANG Modules for Service Assurance";
}
revision 2020-01-13 {
  description
    "Initial revision.";
  reference
    "RFC xxxx: YANG Modules for Service Assurance";
}

identity interface-idty {
  base sain:subservice-idty;
  description
    "Checks whether an interface is healthy.";
}

augment "/sain:subservices/sain:subservice/sain:parameter" {
  when "derived-from-or-self(sain:type, 'interface-idty')";
  description
    "Specify the required parameters for the interface-idty
    subservice type";
  container parameters {
    description
      "Required parameters for the interface-idty subservice
      type";
    leaf device {
      type string;
      mandatory true;
      description
        "Device supporting the interface.";
    }
  }
}
```

```
    leaf interface {  
        type string;  
        mandatory true;  
        description  
            "Name of the interface.";  
    }  
}  
}
```

<CODE ENDS>

6. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

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

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). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

- * /subservices/subservice/type
- * /subservices/subservice/id
- * /subservices/subservice/under-maintenance
- * /subservices/subservice/maintenance-contact

7. IANA Considerations

7.1. The IETF XML Registry

This document registers two URIs in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested:

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

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

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

7.2. The YANG Module Names Registry

This document registers three YANG modules in the YANG Module Names registry [RFC7950]. Following the format in [RFC7950], the the following registrations are requested:

name: ietf-service-assurance
namespace: urn:ietf:params:xml:ns:yang:ietf-service-assurance
prefix: sain
reference: RFC XXXX

name: ietf-service-assurance-device
namespace: urn:ietf:params:xml:ns:yang:ietf-service-assurance-device
prefix: sain-device
reference: RFC XXXX

name: ietf-service-assurance-interface
namespace: urn:ietf:params:xml:ns:yang:ietf-service-assurance-interface
prefix: sain-interface
reference: RFC XXXX

8. Open Issues

-None

9. References

9.1. Normative References

- [I-D.ietf-opsawg-service-assurance-architecture]
Claise, B., Quilbeuf, J., Lopez, D. R., Voyer, D., and T. Arumugam, "Service Assurance for Intent-based Networking Architecture", Work in Progress, Internet-Draft, draft-ietf-opsawg-service-assurance-architecture-03, 7 March 2022, <<https://www.ietf.org/archive/id/draft-ietf-opsawg-service-assurance-architecture-03.txt>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [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>>.
- [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>>.

9.2. Informative 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>>.
- [RFC7895] Bierman, A., Bjorklund, M., and K. Watsen, "YANG Module Library", RFC 7895, DOI 10.17487/RFC7895, June 2016, <<https://www.rfc-editor.org/info/rfc7895>>.
- [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>>.

Appendix A. Vendor-specific Subservice Extension: example-service-assurance-device-acme YANG module

A.1. Tree View

The following tree diagram [RFC8340] provides an overview of the example-service-assurance-device-acme data model.

```
module: example-service-assurance-device-acme

  augment /sain:subservices/sain:subservice/sain:parameter:
    +--rw parameters
      +--rw device                string
      +--rw acme-specific-parameter  string
```

A.2. Complete Tree View

The following tree diagram [RFC8340] provides an overview of the ietf-service-assurance, ietf-service-assurance-device, and example-service-assurance-device-acme data models.

```

module: ietf-service-assurance
  +--ro assurance-graph-version          yang:counter64
  +--ro assurance-graph-last-change      yang:date-and-time
  +--rw subservices
    +--rw subservice* [type id]
      +--rw type                          identityref
      +--rw id                            string
      +--ro last-change?
      |      yang:date-and-time
      +--ro label?                        string
      +--rw under-maintenance?            boolean
      +--rw maintenance-contact           string
      +--rw (parameter)?
        +--:(service-instance-parameter)
        |   +--rw service-instance-parameter
        |   |   +--rw service              string
        |   |   +--rw instance-name        string
        +--:(sain-device:parameters)
        |   +--rw sain-device:parameters
        |   |   +--rw sain-device:device    string
        +--:(example-device-acme:parameters)
        |   +--rw example-device-acme:parameters
        |   |   +--rw example-device-acme:device
        |   |   |   string
        |   |   +--rw example-device-acme:acme-specific-parameter
        |   |   |   string
        +--:(sain-interface:parameters)
        |   +--rw sain-interface:parameters
        |   |   +--rw sain-interface:device    string
        |   |   +--rw sain-interface:interface string
      +--ro health-score?                 union
      +--ro symptoms-history-start?
      |      yang:date-and-time
      +--rw symptoms
        +--ro symptom* [start-date-time id]
          +--ro id                        string
          +--ro health-score-weight?      uint8
          +--ro description?              string
          +--ro start-date-time           yang:date-and-time
          +--ro stop-date-time?           yang:date-and-time
      +--rw dependencies
        +--rw dependency* [type id]
          +--rw type
          |      -> /subservices/subservice/type
          +--rw id                        leafref
          +--rw dependency-type?          identityref

```

A.3. Concepts

Under some circumstances, vendor-specific subservice types might be required. As an example of this vendor-specific implementation, this section shows how to augment the `ietf-service-assurance-device` module to add custom support for the device subservice, specific to the ACME Corporation. The specific version adds a new parameter, named `acme-specific-parameter`.

A.4. YANG Module

```
module example-service-assurance-device-acme {
  yang-version 1.1;
  namespace "urn:example:example-service-assurance-device-acme";
  prefix example-device-acme;

  import ietf-service-assurance {
    prefix sain;
    reference
      "RFC xxxx: YANG Modules for Service Assurance";
  }
  import ietf-service-assurance-device {
    prefix sain-device;
    reference
      "RFC xxxx: YANG Modules for Service Assurance";
  }

  organization
    "IETF OPSAWG 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 extends the ietf-service-assurance-device module to
    add specific support for devices of ACME Corporation.

    ACME Network Device is healthy.

    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 2022-04-07 {
  description
    "Fix mandatory in augment error by moving when clause.
     Shorten prefix.
     Fix module description";
  reference
    "RFC xxxx: YANG Modules for Service Assurance";
}
revision 2021-06-28 {
  description
    "Renamed the parameters container.";
  reference
    "RFC xxxx: YANG Modules for Service Assurance";
}
revision 2020-01-13 {
  description
    "Added the maintenance window concept.";
  reference
    "RFC xxxx: YANG Modules for Service Assurance";
}
revision 2019-11-16 {
  description
    "Initial revision.";
  reference
    "RFC xxxx: YANG Modules for Service Assurance";
}

identity device-acme-idty {
  base sain-device:device-idty;
  description
    "Network Device is healthy.";
}

augment "/sain:subservices/sain:subservice/sain:parameter" {
  when "derived-from-or-self(sain:type, 'device-acme-idty')";
  description
```

```

    "Specify the required parameters for a new subservice type";
  container parameters {
    description
      "Specify the required parameters for the device-acme-idty
      subservice type";
    leaf device {
      type string;
      mandatory true;
      description
        "The device to monitor.";
    }
    leaf acme-specific-parameter {
      type string;
      mandatory true;
      description
        "The ACME Corporation sepcific parameter.";
    }
  }
}
}
}

```

Appendix B. Further Extensions: IP Connectivity and IS-IS subservices

In this section, we provide two additional YANG models to completely cover the example from Figure 2 in [I-D.ietf-opsawg-service-assurance-architecture]. The complete normalization of these modules is to be done in future work.

B.1. IP Connectivity Tree View

That subservice represents the unicast connectivity between two IP addresses located on to different devices. Such a subservice could report symptoms such as "No route found". The following tree diagram [RFC8340] provides an overview of the example-service-assurance-ip-connectivity data model.

```

module: example-service-assurance-ip-connectivity

augment /sain:subservices/sain:subservice/sain:parameter:
  +--rw parameters
    +--rw device1      string
    +--rw address1     inet:ip-address
    +--rw device2      string
    +--rw address2     inet:ip-address

```

To specify the connectivity that we are interested in, we specify two IP addresses and two devices. The subservice assures that the connectivity between IP address 1 on device 1 and IP address 2 on device 2 is healthy.

B.2. IS-IS Tree View

The following tree diagram [RFC8340] provides an overview of the example-service-assurance-is-is data model.

```
module: example-service-assurance-is-is
```

```
  augment /sain:subservices/sain:subservice/sain:parameter:
    +--rw parameters
      +--rw instance-name    string
```

The parameter of this subservice is the name of the IS-IS instance to assure.

B.3. Global Tree View

The following tree diagram [RFC8340] provides an overview of the ietf-service-assurance, ietf-service-assurance-device, example-service-assurance-device-acme, example-service-assurance-ip-connectivity and example-service-assurance-is-is data models.

```
module: ietf-service-assurance
  +--ro assurance-graph-version      yang:counter64
  +--ro assurance-graph-last-change  yang:date-and-time
  +--rw subservices
    +--rw subservice* [type id]
      +--rw type                                identityref
      +--rw id                                string
      +--ro last-change?
        | yang:date-and-time
      +--ro label?                            string
      +--rw under-maintenance?              boolean
      +--rw maintenance-contact             string
      +--rw (parameter)?
        | +--:(service-instance-parameter)
        | | +--rw service-instance-parameter
        | | | +--rw service                string
        | | | +--rw instance-name          string
        | | +--:(example-ip-connectivity:parameters)
        | | | +--rw example-ip-connectivity:parameters
        | | | | +--rw example-ip-connectivity:device1    string
        | | | | +--rw example-ip-connectivity:address1
```



```

|         inet:ip-address
+--rw example-ip-connectivity:device2      string
+--rw example-ip-connectivity:address2
|         inet:ip-address
+--:(example-is-is:parameters)
|   +--rw example-is-is:parameters
|   +--rw example-is-is:instance-name      string
+--:(sain-device:parameters)
|   +--rw sain-device:parameters
|   +--rw sain-device:device               string
+--:(example-device-acme:parameters)
|   +--rw example-device-acme:parameters
|   +--rw example-device-acme:device
|   |   string
|   +--rw example-device-acme:acme-specific-parameter
|   |   string
+--:(sain-interface:parameters)
|   +--rw sain-interface:parameters
|   +--rw sain-interface:device            string
|   +--rw sain-interface:interface        string
+--ro health-score?                        union
+--ro symptoms-history-start?
|   yang:date-and-time
+--rw symptoms
|   +--ro symptom* [start-date-time id]
|   |   +--ro id                        string
|   |   +--ro health-score-weight?    uint8
|   |   +--ro description?            string
|   |   +--ro start-date-time          yang:date-and-time
|   |   +--ro stop-date-time?         yang:date-and-time
+--rw dependencies
|   +--rw dependency* [type id]
|   |   +--rw type
|   |   |   -> /subservices/subservice/type
|   |   +--rw id                      leafref
|   |   +--rw dependency-type?        identityref

```

B.4. IP Connectivity YANG Module

```

module example-service-assurance-ip-connectivity {
  yang-version 1.1;
  namespace "urn:example:example-service-assurance-ip-connectivity";
  prefix example-ip-connectivity;

  import ietf-inet-types {
    prefix inet;
    reference

```

```
    "RFC 6991: Common YANG Data Types";
}
import ietf-service-assurance {
    prefix sain;
    reference
        "RFC xxxx: YANG Modules for Service Assurance";
}

organization
    "IETF OPSAWG 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 example module extends the ietf-service-assurance module to
    add support for the subservice ip-connectivity.

    Checks whether the ip connectivity between two ip addresses
    belonging to two network devices is healthy.

    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 2022-04-07 {
    description
        "Fix mandatory in augment error by moving when clause.
        Shorten prefix. Fix module description.
        Fix module description";
    reference
        "RFC xxxx: YANG Modules for Service Assurance";
```

```
}
revision 2021-06-28 {
  description
    "Initial revision.";
  reference
    "RFC xxxx: YANG Modules for Service Assurance";
}

identity ip-connectivity-idty {
  base sain:subservice-idty;
  description
    "Checks connectivity between two IP addresses.";
}

augment "/sain:subservices/sain:subservice/sain:parameter" {
  when "derived-from-or-self(sain:type, 'ip-connectivity-idty')";
  description
    "Specify the required parameters for the ip-connectivity-idty
    subservice type";
  container parameters {
    description
      "Required parameters for the ip-connectivity-idty
      subservice type";
    leaf device1 {
      type string;
      mandatory true;
      description
        "Device at the first end of the connection.";
    }
    leaf address1 {
      type inet:ip-address;
      mandatory true;
      description
        "Address at the first end of the connection.";
    }
    leaf device2 {
      type string;
      mandatory true;
      description
        "Device at the second end of the connection.";
    }
    leaf address2 {
      type inet:ip-address;
      mandatory true;
      description
        "Address at the second end of the connection.";
    }
  }
}
```

```
}  
}
```

B.5. IS-IS YANG Module

```
module example-service-assurance-is-is {  
  yang-version 1.1;  
  namespace "urn:example:example-service-assurance-is-is";  
  prefix example-is-is;  
  
  import ietf-service-assurance {  
    prefix sain;  
    reference  
      "RFC xxxx: YANG Modules for Service Assurance";  
  }  
  
  organization  
    "IETF OPSAWG 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 extends the ietf-service-assurance module to  
    add support for the subservice is-is.  
  
    Checks whether an IS-IS instance is healthy.  
  
    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 2022-04-07 {
  description
    "Fix mandatory in augment error by moving when clause.
    Shorten prefix. Fix module description.
    Fix module description";
  reference
    "RFC xxxx: YANG Modules for Service Assurance";
}
revision 2021-06-28 {
  description
    "Initial revision.";
  reference
    "RFC xxxx: YANG Modules for Service Assurance";
}

identity is-is-idty {
  base sain:subservice-idty;
  description
    "Health of IS-IS routing protocol.";
}

augment "/sain:subservices/sain:subservice/sain:parameter" {
  when "derived-from-or-self(sain:type, 'is-is-idty')";
  description
    "Specify the required parameters for a new subservice
    type";
  container parameters {
    description
      "Specify the required parameters for the IS-IS subservice
      type";
    leaf instance-name {
      type string;
      mandatory true;
      description
        "The instance to monitor.";
    }
  }
}
}
```

Appendix C. Example of YANG instances

This section contains examples of YANG instances that conform to the YANG modules. The validity of these data instances has been checked using yangson (<https://yangson.labs.nic.cz/>). Yangson requires a YANG library [RFC7895] to define the complete model against which the data instance must be validated. We provide in Appendix D the JSON library file, named "ietf-service-assurance-library.json", that we used for validation.

We provide below the contents of the file "example_configuration_instance.json" which contains the configuration data that models the Figure 2 of [I-D.ietf-opsawg-service-assurance-architecture]. The instance can be validated with yangson by using the invocation "yangson -v example_configuration_instance.json ietf-service-assurance-library.json", assuming all the files (YANG and JSON) defined in this draft reside in the current folder.

```
{
  "ietf-service-assurance:subservices": {
    "subservice": [
      {
        "type": "service-instance-idty",
        "id": "simple-tunnel/example",
        "service-instance-parameter": {
          "service": "simple-tunnel",
          "instance-name": "example"
        },
        "dependencies": {
          "dependency": [
            {
              "type": "ietf-service-assurance-interface:interface-idty",
              "id": "interface/peer1/tunnel0",
              "dependency-type": "impacting-dependency"
            },
            {
              "type": "ietf-service-assurance-interface:interface-idty",
              "id": "interface/peer2/tunnel9",
              "dependency-type": "impacting-dependency"
            },
            {
              "type":
                "example-service-assurance-ip-connectivity:ip-connectivity-idty",
              "id": "connectivity/peer1/2001:db8::1/peer2/2001:db8::2",
              "dependency-type": "impacting-dependency"
            }
          ]
        }
      }
    ]
  }
}
```

```
    }
  },
  {
    "type":
    "example-service-assurance-ip-connectivity:ip-connectivity-idty",
    "id": "connectivity/peer1/2001:db8::1/peer2/2001:db8::2",
    "example-service-assurance-ip-connectivity:parameters": {
      "device1": "Peer1",
      "address1": "2001:db8::1",
      "device2": "Peer2",
      "address2": "2001:db8::2"
    },
    "dependencies": {
      "dependency": [
        {
          "type": "ietf-service-assurance-interface:interface-idty",
          "id": "interface/peer1/physical0",
          "dependency-type": "impacting-dependency"
        },
        {
          "type": "ietf-service-assurance-interface:interface-idty",
          "id": "interface/peer2/physical5",
          "dependency-type": "impacting-dependency"
        },
        {
          "type": "example-service-assurance-is-is:is-is-idty",
          "id": "is-is/instance1",
          "dependency-type": "impacting-dependency"
        }
      ]
    }
  },
  {
    "type": "example-service-assurance-is-is:is-is-idty",
    "id": "is-is/instance1",
    "example-service-assurance-is-is:parameters": {
      "instance-name": "instance1"
    }
  },
  {
    "type": "ietf-service-assurance-interface:interface-idty",
    "id": "interface/peer1/tunnel0",
    "ietf-service-assurance-interface:parameters": {
      "device": "Peer1",
      "interface": "tunnel0"
    },
    "dependencies": {
      "dependency": [
```

```
        {
            "type": "ietf-service-assurance-interface:interface-idty",
            "id": "interface/peer1/physical0",
            "dependency-type": "impacting-dependency"
        }
    ]
}
},
{
    "type": "ietf-service-assurance-interface:interface-idty",
    "id": "interface/peer1/physical0",
    "ietf-service-assurance-interface:parameters": {
        "device": "Peer1",
        "interface": "physical0"
    },
    "dependencies": {
        "dependency": [
            {
                "type": "ietf-service-assurance-device:device-idty",
                "id": "interface/peer1",
                "dependency-type": "impacting-dependency"
            }
        ]
    }
},
{
    "type": "ietf-service-assurance-device:device-idty",
    "id": "interface/peer1",
    "ietf-service-assurance-device:parameters": {
        "device": "Peer1"
    }
},
{
    "type": "ietf-service-assurance-interface:interface-idty",
    "id": "interface/peer2/tunnel9",
    "ietf-service-assurance-interface:parameters": {
        "device": "Peer2",
        "interface": "tunnel9"
    },
    "dependencies": {
        "dependency": [
            {
                "type": "ietf-service-assurance-interface:interface-idty",
                "id": "interface/peer2/physical15",
                "dependency-type": "impacting-dependency"
            }
        ]
    }
}
```



```

    },
    {
      "type": "ietf-service-assurance-interface:interface-idty",
      "id": "interface/peer2/physical5",
      "ietf-service-assurance-interface:parameters": {
        "device": "Peer2",
        "interface": "physical5"
      },
      "dependencies": {
        "dependency": [
          {
            "type": "ietf-service-assurance-device:device-idty",
            "id": "interface/peer2",
            "dependency-type": "impacting-dependency"
          }
        ]
      }
    },
    {
      "type": "ietf-service-assurance-device:device-idty",
      "id": "interface/peer2",
      "ietf-service-assurance-device:parameters": {
        "device": "Peer2"
      }
    }
  ]
}

```

Appendix D. YANG Library for Service Assurance

This section provides the JSON encoding of the YANG library [RFC7895] listing all modules defined in this draft and their dependencies. This library can be used to validate data instances using yangson, as explained in the previous section.

```

{
  "ietf-yang-library:modules-state": {
    "module-set-id": "ietf-service-assurance@2022-04-07",
    "module": [
      {
        "name": "ietf-service-assurance",
        "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-service-assurance",
        "revision": "2022-04-07",
        "conformance-type": "implement"
      }
    ]
  }
}

```

```
    },
    {
      "name": "ietf-service-assurance-device",
      "namespace":
        "urn:ietf:params:xml:ns:yang:ietf-service-assurance-device",
      "revision": "2022-04-07",
      "conformance-type": "implement"
    },
    {
      "name": "ietf-service-assurance-interface",
      "namespace":
        "urn:ietf:params:xml:ns:yang:ietf-service-assurance-interface",
      "revision": "2022-04-07",
      "conformance-type": "implement"
    },
    {
      "name": "example-service-assurance-device-acme",
      "namespace":
        "urn:example:example-service-assurance-device-acme",
      "revision": "2022-04-07",
      "conformance-type": "implement"
    },
    {
      "name": "example-service-assurance-is-is",
      "namespace": "urn:example:example-service-assurance-is-is",
      "revision": "2022-04-07",
      "conformance-type": "implement"
    },
    {
      "name": "example-service-assurance-ip-connectivity",
      "namespace":
        "urn:example:example-service-assurance-ip-connectivity",
      "revision": "2022-04-07",
      "conformance-type": "implement"
    },
    {
      "name": "ietf-yang-types",
      "namespace": "urn:ietf:params:xml:ns:yang:ietf-yang-types",
      "revision": "2021-04-14",
      "conformance-type": "import"
    },
    {
      "name": "ietf-inet-types",
      "namespace": "urn:ietf:params:xml:ns:yang:ietf-inet-types",
      "revision": "2021-02-22",
      "conformance-type": "import"
    }
  ]
}
```

```
}  
}
```

Appendix E. Changes between revisions

v04 - v05

- * Remove Guidelines section
- * Move informative parts (examples) to appendix
- * Minor text edits and reformulations

v03 - v04

- * Fix YANG errors
- * Change is-is and ip-connectivity subservices from ietf to example.
- * Mention that models are NMDA compliant
- * Fix typos, reformulate for clarity

v02 - v03

- * Change counter32 to counter64 to avoid resetting too frequently
- * Explain why relation between health-score and symptom's health-score-weight is not defined and how it could be defined

v01 - v02

- * Explicitly represent the fact that the health-score could not be computed (value -1)

v00 - v01

- * Added needed subservice to model example from architecture draft
- * Added guideline section for naming models
- * Added data instance examples and validation procedure
- * Added the "parameters" container in the interface YANG module to correct a bug.

Acknowledgements

The authors would like to thank Jan Lindblad for his help during the design of these YANG modules. The authors would like to thank Stephane Litkowski, Charles Eckel, Mohamed Boucadair and Tom Petch for their reviews.

Authors' Addresses

Benoit Claise
Huawei
Email: benoit.claise@huawei.com

Jean Quilbeuf
Huawei
Email: jean.quilbeuf@huawei.com

Paolo Lucente
NTT
Siriusdreef 70-72
2132 Hoofddorp
Netherlands
Email: paolo@ntt.net

Paolo Fasano
TIM S.p.A
via G. Reiss Romoli, 274
10148 Torino
Italy
Email: paolo2.fasano@telecomitalia.it

Thangam Arumugam
Cisco Systems, Inc.
Milpitas (California),
United States
Email: tarumuga@cisco.com

OPSAWG Working Group
Internet-Draft
Intended status: Standards Track
Expires: 6 November 2022

B. Wu, Ed.
Q. Wu, Ed.
Huawei
M. Boucadair, Ed.
Orange
O. Gonzalez de Dios
Telefonica
B. Wen
Comcast
5 May 2022

A YANG Model for Network and VPN Service Performance Monitoring
draft-ietf-opsawg-yang-vpn-service-pm-08

Abstract

The data model for network topologies defined in RFC 8345 introduces vertical layering relationships between networks that can be augmented to cover network and service topologies. This document defines a YANG module for performance monitoring (PM) of both networks and VPN services that can be used to monitor and manage network performance on the topology at higher layer or the service topology between VPN sites.

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 6 November 2022.

Copyright Notice

Copyright (c) 2022 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	2
2. Terminology	3
2.1. Acronyms	3
3. Network and VPN Service Performance Monitoring Model Usage .	4
3.1. Collecting Data via Pub/Sub Mechanism	5
3.2. Collecting Data On-demand	6
4. Description of The Data Model	6
4.1. Layering Relationship between Multiple Layers of Topology	6
4.2. Network Level	9
4.3. Node Level	9
4.4. Link and Termination Point Level	10
5. Network and VPN Service Performance Monitoring YANG Module .	14
6. Security Considerations	29
7. IANA Considerations	30
8. Acknowledgements	31
9. Contributors	31
10. References	31
10.1. Normative References	31
10.2. Informative References	34
Appendix A. Illustrative Examples	35
A.1. VPN Performance Subscription Example	35
A.2. Example of VPN Performance Snapshot	37
A.3. Example of Percentile Monitoring	39
Authors' Addresses	39

1. Introduction

[RFC8969] describes a framework for automating service and network management with YANG [RFC6020] models. It defines that the performance measurement telemetry model should be tied to the services (such as a Layer 3 VPN or Layer 2 VPN) or to the network models to monitor the overall network performance and the Service Level Agreements (SLAs).

The performance of VPN services is associated with the performance changes of the underlay network that carries VPN services, such as the delay of the underlay tunnels and the packet loss status of the device interfaces. Additionally, the integration of Layer 2/Layer 3 VPN performance and network performance data enables the orchestrator to subscribe to VPN service performance in a unified manner. Therefore, this document defines a YANG module for both network and VPN service performance monitoring (PM). The module can be used to monitor and manage network performance on the topology level or the service topology between VPN sites, in particular.

This document does not introduce new metrics for network performance or mechanisms for measuring network performance, but uses the existing mechanisms and statistics to display the performance monitoring statistics at the network and service layers. All these metrics are defined as unidirectional metrics.

The YANG module defined in this document is designed as an augmentation to the network topology YANG model defined in [RFC8345] and draws on relevant YANG types defined in [RFC6991], [RFC8345], [RFC8532], and [RFC9181].

Appendix A provides a set of examples to illustrate the use of the module.

2. Terminology

The following terms are defined in [RFC7950] and are used in this specification:

- * augment
- * data model
- * data node

The terminology for describing YANG data models is found in [RFC7950].

The tree diagrams used in this document follow the notation defined in [RFC8340].

2.1. Acronyms

The following acronyms are used in the document:

L2VPN Layer 2 Virtual Private Network
L3VPN Layer 3 Virtual Private Network

L2NM	L2VPN Network Model
L3NM	L3VPN Network Model
MPLS	Multiprotocol Label Switching
OAM	Operations, Administration, and Maintenance
OWAMP	One-Way Active Measurement Protocol
PE	Provider Edge
PM	Performance Monitoring
SLA	Service Level Agreement
TP	Termination Point, as defined in [RFC8345] section 4.2
TWAMP	Two-Way Active Measurement Protocol
VPLS	Virtual Private LAN Service
VPN	Virtual Private Network

3. Network and VPN Service Performance Monitoring Model Usage

Models are key for automating network management operations. According to [RFC8969], together with service and network models, performance measurement telemetry models are needed to monitor network performance to meet specific service requirements (typically captured in an SLA).

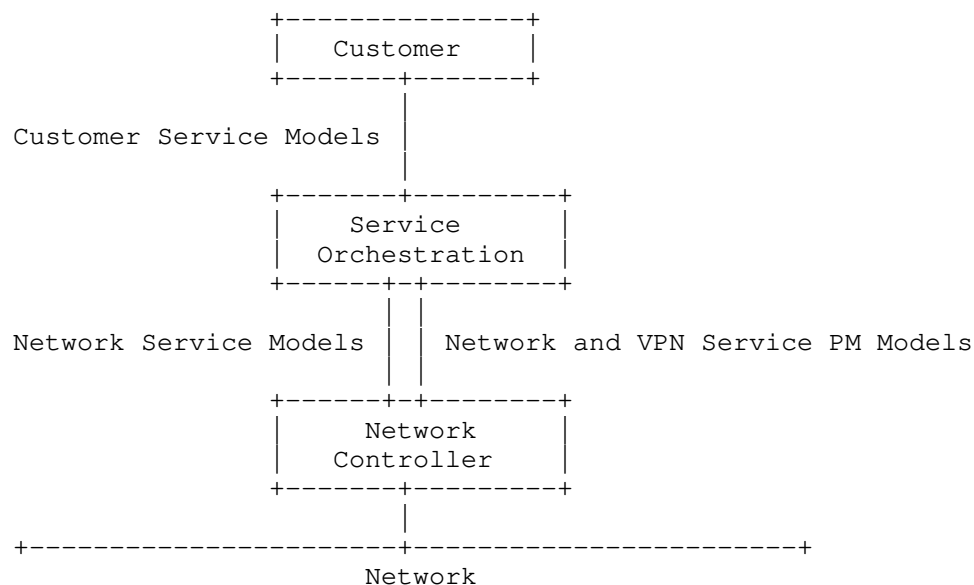


Figure 1: Reference Architecture

As shown in Figure 1, in the context of the layering model architecture described in [RFC8309], the network and VPN service performance monitoring (PM) model can be used to expose a set of

performance information to the above layer. Such information can be used by an orchestrator to subscribe to performance data. The network controller will then notify the orchestrator about corresponding parameter changes.

Before using the model, the controller needs to establish complete topology visibility of the network and VPN. For example, the controller can use network information from [RFC8345], [I-D.ietf-opsawg-sap] or VPN information from [RFC9182], [I-D.ietf-opsawg-l2nm]. Then the controller derives network or VPN level performance data by aggregating (and filtering) lower-level data collected via monitoring counters of the involved devices.

The network or VPN performance data can be based on different sources. For example, the performance monitoring data per link in the underlying network can be collected using a network performance measurement method such as One-Way Active Measurement Protocol (OWAMP) [RFC4656], Two-Way Active Measurement Protocol (TWAMP) [RFC5357], Simple Two-way Active Measurement Protocol (STAMP) [RFC8762], and Multiprotocol Label Switching (MPLS) Loss and Delay Measurement [RFC6374]. The performance monitoring information reflecting the quality of the network or VPN service (e.g., end-to-end network performance data between source node and destination node in the network or between VPN sites) can be computed and aggregated, for example, using the information from the Traffic Engineering Database (TED), [RFC7471] [RFC8570] [RFC8571] or LMAP [RFC8194].

The measurement and report intervals that are associated with these performance data usually depend on the configuration of the specific measurement method or collection method or various combinations. This document defines a network-wide measurement interval to align measurement requirements for networks or VPN services.

In addition, the amount of performance data collected from the devices can be huge. To avoid receiving a large amount of operational data of VPN instances, VPN interfaces, or tunnels, the network controller can specifically subscribe to metric-specific data using the tagging methods defined in [I-D.ietf-netmod-node-tags].

3.1. Collecting Data via Pub/Sub Mechanism

Some applications such as service-assurance applications, which must maintain a continuous view of operational data and state, can use the subscription model specified in [RFC8641] to subscribe to the specific network performance data or VPN service performance data they are interested in, at the data source. For example, network or VPN topology updates may be obtained through on-change notifications [RFC8641]. For dynamic PM data, various notifications can be

specified to obtain more complete data. A periodic notification [RFC8641] can be specified to obtain real-time performance data, a replay notification defined in [RFC5277] or [RFC8639] can be specified to obtain historical data, or alarm notifications [RFC8632] can be specified to get alarms for the metrics which exceed or fall below the performance threshold.

The data source can, then, use the network and VPN service assurance model defined in this document and the YANG Push model [RFC8641] to distribute specific telemetry data to target recipients.

3.2. Collecting Data On-demand

To obtain a snapshot of a large amount of performance data from a network topology or VPN network, service-assurance applications may retrieve information using the network and VPN service PM model through a NETCONF [RFC6241] or a RESTCONF [RFC8040] interface. For example, a specified "link-id" of a VPN can be used as a filter in a RESTCONF GET request to retrieve per-link VPN PM data.

4. Description of The Data Model

This document defines the YANG module, "ietf-network-vpn-pm", which is an augmentation to the "ietf-network" and "ietf-network-topology" modules.

The performance monitoring data augments the service topology as shown in Figure 2.

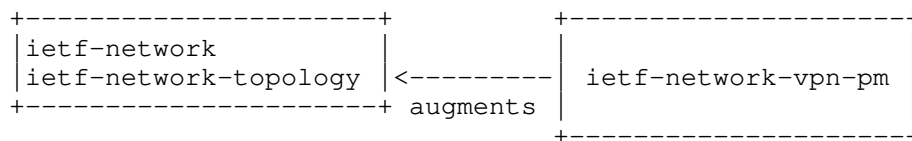


Figure 2: Module Augmentation

4.1. Layering Relationship between Multiple Layers of Topology

[RFC8345] defines a YANG data model for network/service topologies and inventories. The service topology described in [RFC8345] includes the virtual topology for a service layer above Layer 1 (L1), Layer 2 (L2), and Layer 3 (L3). This service topology has the generic topology elements of node, link, and terminating point. One typical example of a service topology is described in Figure 3 of [RFC8345]: two VPN service topologies instantiated over a common L3 topology. Each VPN service topology is mapped onto a subset of nodes from the common L3 topology.

Figure 3 illustrates an example of a topology that maps between the VPN service topology and an underlying network:

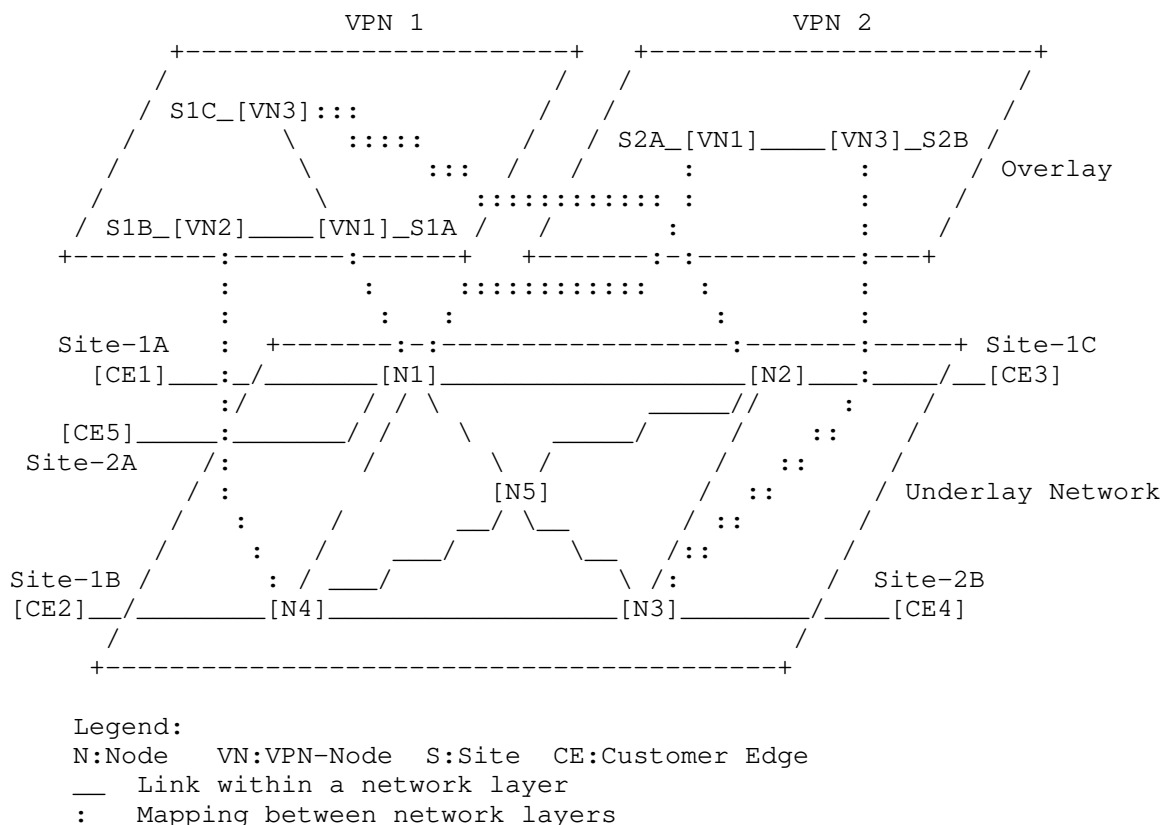


Figure 3: Example of Topology Mapping Between VPN Service Topology and Underlying Network

As shown in Figure 3, two VPN services topologies are built on top of one common underlying physical network:

VPN 1: This service topology supports hub-spoke communications for 'customer 1' connecting the customer's access at three sites: 'Site-1A', 'Site-1B', and 'Site-1C'. These sites are connected to nodes that are mapped to node 1 (N1), node 2 (N2), and node 4 (N4) in the underlying physical network. 'Site-1A' plays the role of hub while 'Site-1B' and 'Site-1C' are configured as spoke.

VPN 2: This service supports any-to-any communications for 'customer

2' connecting the customer's access at two sites: 'Site-2A' and 'Site-2B'. These sites are connected to nodes that are mapped to nodes 1 (N1) and node 3 (N3)5 in the underlying physical network. 'Site-2A' and 'Site-2B' have 'any-to-any' role.

Apart from the association between the VPN topology and the underlay topology, VPN Network PM can also provide the performance status of the underlay network and VPN services. For example, network PM can provide link PM statistics and port statistics. VPN PM can provide statistics on VPN access interfaces, the number of current VRF routes or L2VPN MAC entry of VPN nodes, and performance statistics on the logical point-to-point link between source and destination VPN nodes or between source and destination VPN access interfaces. Figure 4 illustrates an example of VPN PM and the difference between two VPN PM measurement methods. One is the VPN tunnel PM and the other is inter-VPN-access interface PM.

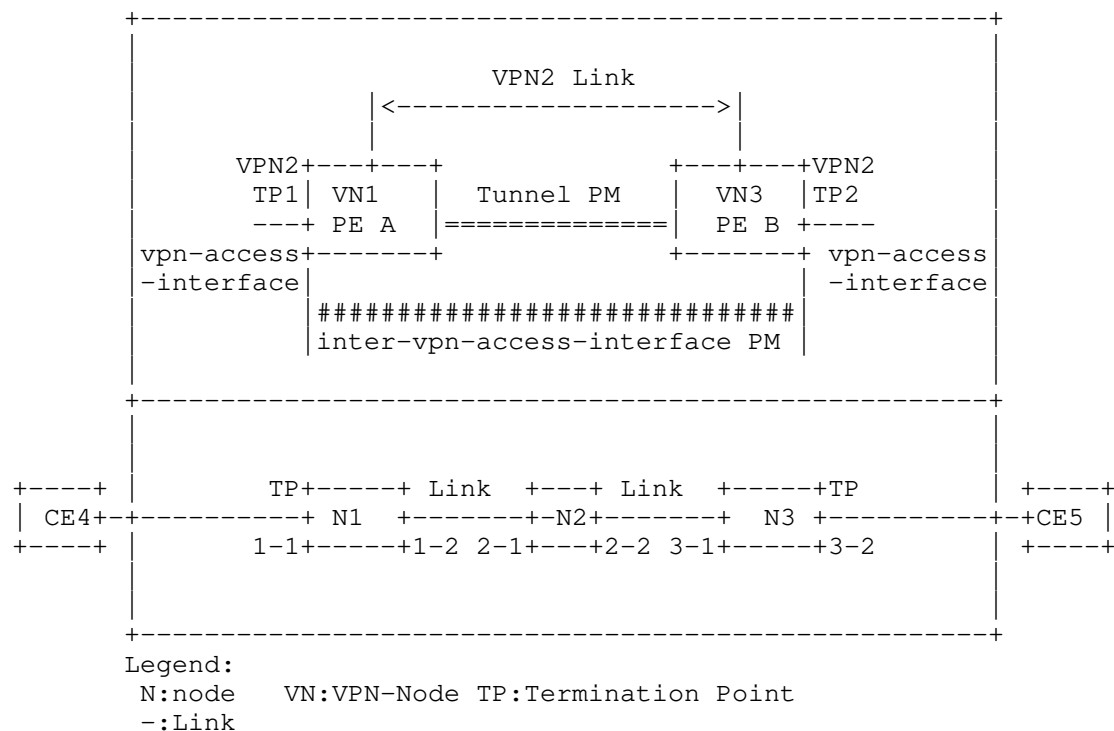


Figure 4: An Example of VPN PM

4.2. Network Level

For network performance monitoring, the container of "networks" in [RFC8345] does not need to be extended.

For VPN service performance monitoring, the container "service-type" is defined to indicate the VPN type, e.g., L3VPN or Virtual Private LAN Service (VPLS). The values are taken from [RFC9181]. When a network topology instance contains the L3VPN or other L2VPN network type, it represents a VPN instance that can perform performance monitoring.

The tree in Figure 5 is a part of ietf-network-vpn-pm tree. It defines the following set of network level attributes:

"vpn-id": Refers to an identifier of VPN service defined in [RFC9181]. This identifier is used to correlate the performance status with the network service configuration.

"vpn-service-topology": Indicates the type of the VPN topology. This model supports "any-to-any", "Hub and Spoke" (where Hubs can exchange traffic), and "Hub and Spoke disjoint" (where Hubs cannot exchange traffic) that are taken from [RFC9181]. These VPN topology types can be used to describe how VPN sites communicate with each other.

```
module: ietf-network-vpn-pm
  augment /nw:networks/nw:network/nw:network-types:
    +--rw service-type!
      +--rw service-type? identityref
  augment /nw:networks/nw:network:
    +--rw vpn-pm-attributes
      +--rw vpn-id?          vpn-common:vpn-id
      +--rw vpn-service-topology? identityref
```

Figure 5: Network Level YANG Tree of the Hierarchies

4.3. Node Level

The tree in Figure 6 is the node part of ietf-network-vpn-pm tree.

For network performance monitoring, a container of "pm-attributes" is augmented to the list of "node" that are defined in [RFC8345]. The container includes the following attributes:

"node-type": Indicates the device type of Provider Edge (PE),

Provider (P) device, or Autonomous System Border Router (ASBR) as defined in [RFC4026] and [RFC4364], so that the performance metric between any two nodes each with specific node type can be reported.

"entry-summary": Lists a set of IPv4 statistics, IPv6 statistics, and MAC statistics. The detailed statistics are specified separately.

For VPN service performance monitoring, the model defines one attribute:

"role": Defines the role in a particular VPN service topology. The roles are taken from [RFC9181] (e.g., any-to-any-role, spoke-role, hub-role).

```
augment /nw:networks/nw:network/nw:node:
  +--rw pm-attributes
    +--rw node-type?          identityref
    +--ro entry-summary
      +--ro ipv4-num
        +--ro maximum-routes?    uint32
        +--ro total-active-routes? uint32
      +--ro ipv6-num
        +--ro maximum-routes?    uint32
        +--ro total-active-routes? uint32
      +--ro mac-num
        +--ro mac-num-limit?      uint32
        +--ro total-active-mac-num? uint32
    +--rw role?                identityref
```

Figure 6: Node Level YANG Tree of the Hierarchies

4.4. Link and Termination Point Level

The tree in Figure 7 is the link and termination point (TP) part of ietf-network-vpn-pm tree.

The 'links' are classified into two types: topology link defined in [RFC8345] and abstract link of a VPN between PEs defined in this module.

The performance data of a link is a collection of counters and gauges that report the performance status.

```

augment /nw:networks/nw:network/nt:link:
  +---rw pm-attributes
    +---rw low-percentile?                percentile
    +---rw intermediate-percentile?       percentile
    +---rw high-percentile?              percentile
    +---rw measurement-interval?         uint32
    +---ro start-time?                   yang:date-and-time
    +---ro end-time?                     yang:date-and-time
    +---ro pm-source?                     identityref
    +---ro one-way-pm-statistics
      +---ro loss-statistics
        +---ro packet-loss-count?        yang:counter64
        +---ro loss-ratio?                percentage
      +---ro delay-statistics
        +---ro unit-value?                identityref
        +---ro min-delay-value?           yang:gauge64
        +---ro max-delay-value?           yang:gauge64
        +---ro low-delay-percentile?      yang:gauge64
        +---ro intermediate-delay-percentile? yang:gauge64
        +---ro high-delay-percentile?     yang:gauge64
      +---ro jitter-statistics
        +---ro unit-value?                identityref
        +---ro min-jitter-value?          yang:gauge64
        +---ro max-jitter-value?          yang:gauge64
        +---ro low-jitter-percentile?     yang:gauge64
        +---ro intermediate-jitter-percentile? yang:gauge64
        +---ro high-jitter-percentile?    yang:gauge64
    +---ro one-way-pm-statistics-per-class* [class-id]
      +---ro class-id                     string
      +---ro loss-statistics
        +---ro packet-loss-count?        yang:counter64
        +---ro loss-ratio?                percentage
      +---ro delay-statistics
        +---ro unit-value?                identityref
        +---ro min-delay-value?           yang:gauge64
        +---ro max-delay-value?           yang:gauge64
        +---ro low-delay-percentile?      yang:gauge64
        +---ro intermediate-delay-percentile? yang:gauge64
        +---ro high-delay-percentile?     yang:gauge64
      +---ro jitter-statistics
        +---ro unit-value?                identityref
        +---ro min-jitter-value?          yang:gauge64
        +---ro max-jitter-value?          yang:gauge64
        +---ro low-jitter-percentile?     yang:gauge64
        +---ro intermediate-jitter-percentile? yang:gauge64
        +---ro high-jitter-percentile?    yang:gauge64
  +---rw vpn-pm-type
    +---rw inter-vpn-access-interface

```

```

    |   +--rw inter-vpn-access-interface?   empty
    +--rw underlay-tunnel!
        +--ro vpn-underlay-transport-type?   identityref
augment /nw:networks/nw:network/nw:node/nt:termination-point:
+--ro pm-statistics
+--ro reference-time?           yang:date-and-time
+--ro inbound-octets?           yang:counter64
+--ro inbound-unicast?         yang:counter64
+--ro inbound-nunicast?        yang:counter64
+--ro inbound-discards?        yang:counter64
+--ro inbound-errors?          yang:counter64
+--ro inbound-unknown-protocol? yang:counter64
+--ro outbound-octets?          yang:counter64
+--ro outbound-unicast?         yang:counter64
+--ro outbound-nunicast?       yang:counter64
+--ro outbound-discards?       yang:counter64
+--ro outbound-errors?         yang:counter64
+--ro vpn-network-access* [network-access-id]
+--ro network-access-id        vpn-common:vpn-id
+--ro reference-time?          yang:date-and-time
+--ro inbound-octets?          yang:counter64
+--ro inbound-unicast?         yang:counter64
+--ro inbound-nunicast?        yang:counter64
+--ro inbound-discards?        yang:counter64
+--ro inbound-errors?          yang:counter64
+--ro inbound-unknown-protocol? yang:counter64
+--ro outbound-octets?          yang:counter64
+--ro outbound-unicast?         yang:counter64
+--ro outbound-nunicast?       yang:counter64
+--ro outbound-discards?       yang:counter64
+--ro outbound-errors?         yang:counter64

```

Figure 7: Link and Termination point Level YANG Tree of the hierarchies

For the data nodes of 'link' depicted in Figure 7, the YANG module defines the following minimal set of link-level performance attributes:

Percentile parameters: The module supports reporting delay and jitter metric by percentile values. By default, low percentile (10th percentile), intermediate percentile (50th percentile), high percentile (90th percentile) are used. Setting a percentile to 0.00 indicates the client is not interested in receiving particular percentile. If all percentile nodes are set to 0.00, this represents that no percentile related nodes will be reported for a given performance metric (e.g., one-way delay, one-way delay variation) and only peak/min values will be reported. For

example, a client can inform the server that it is interested in receiving only high percentiles. Then for a given link, at a given "start-time", "end-time" and "measurement-interval", the 'high-delay-percentile' and 'high-jitter-percentile' will be reported. An example to illustrate the use of percentiles is provided in Appendix A.3.

PM source ("pm-source"): Indicates the performance monitoring source. The data for the topology link can be based, e.g., on BGP-LS [RFC8571]. The statistics of the VPN abstract links can be collected based upon VPN OAM mechanisms, e.g., OAM mechanisms referenced in [RFC9182], or Ethernet service OAM [ITU-T-Y-1731] referenced in [I-D.ietf-opsawg-l2nm]. Alternatively, the data can be based upon the underlay technology OAM mechanisms, for example, Generic Routing Encapsulation (GRE) tunnel OAM.

Measurement interval ("measurement-interval"): Specifies the performance measurement interval, in seconds.

Start time ("start-time"): Indicates the start time of the performance measurement for link statistics.

End time ("end-time"): Indicates the end time of the performance measurement for link statistics.

Reference time ("reference-time"): Indicates the timestamp when the counters are obtained.

Loss statistics: A set of one-way loss statistics attributes that are used to measure end to end loss between VPN sites or between any two network nodes. The exact loss value or the loss percentage can be reported.

Delay statistics: A set of one-way delay statistics attributes that are used to measure end to end latency between VPN sites or between any two network nodes. The peak/min values or percentile values can be reported.

Jitter statistics: A set of one-way IP Packet Delay Variation [RFC3393] statistics attributes that are used to measure end to end jitter between VPN sites or between any two network nodes. The peak/min values or percentile values can be reported.

PM statistics per class ("one-way-pm-statistics-per-class"): Lists p

formance measurement statistics for the topology link or the abstract underlay link between VPN PEs with given "class-id" names. The list is defined separately from "one-way-pm-statistics", which is used to collect generic metrics for unspecified "class-id" names.

VPN PM type ("vpn-pm-type"): Indicates the VPN performance type, which can be inter-vpn-access-interface PM or VPN underlay-tunnel PM. These two methods are common VPN measurement methods. The inter-VPN-access-interface PM is to monitor the performance of logical point-to-point VPN connections between a source and a destination VPN access interfaces. And the underlay-tunnel PM is to monitor the performance of underlay tunnels of VPNs. The inter-VPN-access-interface PM includes PE-PE monitoring. Therefore, usually only one of the two methods is used. The inter-VPN-access-interface PM is defined as an empty leaf, which is not bound to a specific VPN access interface. The source or destination VPN access interface of the measurement can be augmented as needed.

VPN underlay transport type ("vpn-underlay-transport-type"): Indicates the abstract link protocol-type of a VPN, such as GRE or IP-in-IP. The leaf refers to an identifier of the "underlay-transport" defined in [RFC9181], which describes the transport technology to carry the traffic of the VPN service.

For the data nodes of 'termination-point' depicted in Figure 7, the module defines the following minimal set of statistics:

Inbound statistics: A set of inbound statistics attributes that are used to measure the inbound statistics of the termination point, such as received packets, received packets with errors, etc.

Outbound statistics: A set of outbound statistics attributes that are used to measure the outbound statistics of the termination point, such as sent packets, packets that could not be sent due to errors, etc.

VPN network access ("vpn-network-access"): Lists counters of the VPN network access defined in [RFC9182] or [I-D.ietf-opsawg-l2nm]. When multiple VPN network accesses are created using the same physical port, finer-grained metrics can be monitored. If a TP is associated with only a single VPN, this list is not required.

5. Network and VPN Service Performance Monitoring YANG Module

The "ietf-network-vpn-pm" module uses types defined in [RFC8345], [RFC6991], [RFC8532], and [RFC9181].

```
<CODE BEGINS> file "ietf-network-vpn-pm@2022-05-05.yang"
module ietf-network-vpn-pm {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-network-vpn-pm";
  prefix nvp;

  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991: Common YANG Types";
  }
  import ietf-vpn-common {
    prefix vpn-common;
    reference
      "RFC 9181: A Common YANG Data Model for Layer 2 and
      Layer 3 VPNs.";
  }
  import ietf-network {
    prefix nw;
    reference
      "RFC 8345: A YANG Data Model for Network
      Topologies, Section 6.1";
  }
  import ietf-network-topology {
    prefix nt;
    reference
      "RFC 8345: A YANG Data Model for Network
      Topologies, Section 6.2";
  }
  import ietf-lime-time-types {
    prefix lime;
    reference
      "RFC 8532: Generic YANG Data Model for the Management of
      Operations, Administration, and Maintenance (OAM) Protocols
      That Use Connectionless Communications";
  }

  organization
    "IETF OPSAWG (Operations and Management Area Working Group)";
  contact
    "WG Web:  <https://datatracker.ietf.org/wg/opsawg/>
    WG List:  <mailto:opsawg@ietf.org>

    Editor: Bo Wu
           <lane.wubo@huawei.com>
    Editor: Mohamed Boucadair
           <mohamed.boucadair@orange.com>
    Editor: Qin Wu
```

```
<bill.wu@huawei.com>
Author: Oscar Gonzalez de Dios
       <oscar.gonzalezdedios@telefonica.com>
Author: Bin Wen
       <bin_wen@comcast.com>;
description
  "This module defines a model for Network and VPN Service
  Performance monitoring.

  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
  (https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself
  for full legal notices.";

// RFC Ed.: update the date below with the date of RFC
// publication and remove this note.
// RFC Ed.: replace XXXX with actual RFC number and remove
// this note.

revision 2022-05-05 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Model for Network and VPN Service
    Performance Monitoring";
}

identity node-type {
  description
    "Base identity for node type";
}

identity pe {
  base node-type;
  description
    "Provider Edge (PE) node type. A PE is the name of the device
    or set of devices at the edge of the provider network with the
    functionality that is needed to interface with the customer.";
}
```

```
identity p {
  base node-type;
  description
    "Provider router node type. That is, a router
     in the core network that does not have interfaces
     directly toward a customer.";
}

identity asbr {
  base node-type;
  description
    "Autonomous System Border Router (ASBR) node type.";
  reference
    "RFC 4364: BGP/MPLS IP Virtual Private Networks (VPNs)";
}

identity pm-source-type {
  description
    "Base identity from which specific performance monitoring
     mechanism types are derived.";
}

identity pm-source-bgppls {
  base pm-source-type;
  description
    "Indicates BGP-LS as the performance monitoring metric source";
  reference
    "RFC 8571: BGP - Link State (BGP-LS) Advertisement of
     IGP Traffic Engineering Performance Metric Extensions";
}

identity pm-source-owamp {
  base pm-source-type;
  description
    "Indicates One-Way Active Measurement Protocol (OWAMP)
     as the performance monitoring metric source.";
  reference
    "RFC 4656: A One-Way Active Measurement Protocol (OWAMP)";
}

identity pm-source-twamp {
  base pm-source-type;
  description
    "Indicates Two-Way Active Measurement Protocol (TWAMP)
     as the performance monitoring metric source.";
  reference
    "RFC 5357: A Two-Way Active Measurement Protocol (TWAMP)";
}
```

```
identity pm-source-stamp {
  base pm-source-type;
  description
    "Indicates Simple Two-way Active Measurement Protocol (STAMP)
    as the performance monitoring metric source.";
  reference
    "RFC 8762: Simple Two-Way Active Measurement Protocol";
}

identity pm-source-y-1731 {
  base pm-source-type;
  description
    "Indicates Ethernet OAM Y.1731 as the performance monitoring
    metric source.";
  reference
    "ITU-T Y.1731: Operations, administration and
    maintenance (OAM) functions and mechanisms
    for Ethernet-based networks";
}

typedef percentage {
  type decimal64 {
    fraction-digits 5;
    range "0..100";
  }
  description
    "Percentage.";
}

typedef percentile {
  type decimal64 {
    fraction-digits 2;
    range "0..100";
  }
  description
    "The percentile is a value between 0 and 100,
    e.g. 10.00, 99.90 ,99.99 etc..
    For example, for a given one-way delay measurement,
    if the percentile is set to 95.00 and
    the 95th percentile one-way delay is 2 milliseconds,
    then the 95 percent of the sample value
    is less than or equal to 2 milliseconds.";
}

grouping entry-summary {
  description
    "Entry summary grouping used for network topology
    augmentation.";
```

```
container entry-summary {
  config false;
  description
    "Container for VPN or network entry summary.";
  container ipv4-num {
    leaf maximum-routes {
      type uint32;
      description
        "Indicates the maximum number of IPv4 routes
        for the VPN.";
    }
    leaf total-active-routes {
      type uint32;
      description
        "Indicates total active IPv4 routes for the VPN.";
    }
    description
      "IPv4-specific parameters.";
  }
  container ipv6-num {
    leaf maximum-routes {
      type uint32;
      description
        "Indicates the maximum number of IPv6 routes
        for the VPN.";
    }
    leaf total-active-routes {
      type uint32;
      description
        "Indicates total active IPv6 routes for the VPN.";
    }
    description
      "IPv6-specific parameters.";
  }
  container mac-num {
    leaf mac-num-limit {
      type uint32;
      description
        "Maximum number of MAC addresses.";
    }
    leaf total-active-mac-num {
      type uint32;
      description
        "Total active MAC entries for the VPN.";
    }
    description
      "MAC statistics.";
  }
}
```

```
    }  
  }  
  
  grouping link-loss-statistics {  
    description  
      "Grouping for per link error statistics.";  
    container loss-statistics {  
      description  
        "One-way link loss summarized information.";  
      reference  
        "RFC 4656: A One-way Active Measurement Protocol (OWAMP)  
        ITU-T Y.1731: Operations, administration and  
        maintenance (OAM) functions and mechanisms  
        for Ethernet-based networks";  
      leaf packet-loss-count {  
        type yang:counter64;  
        description  
          "Total received packet drops count.";  
      }  
      leaf loss-ratio {  
        type percentage;  
        description  
          "Loss ratio of the packets. Express as percentage  
          of packets lost with respect to packets sent.";  
      }  
    }  
  }  
}  
  
grouping link-delay-statistics {  
  description  
    "Grouping for per link delay statistics.";  
  container delay-statistics {  
    description  
      "One-way link delay summarized information.";  
    reference  
      "RFC 4656: A One-way Active Measurement Protocol (OWAMP)  
      ITU-T Y.1731: Operations, administration and  
      maintenance (OAM) functions and mechanisms  
      for Ethernet-based networks";  
    leaf unit-value {  
      type identityref {  
        base lime:time-unit-type;  
      }  
      default "lime:milliseconds";  
      description  
        "Time units, where the options are s, ms, ns, etc.";  
    }  
    leaf min-delay-value {
```



```
        type yang:gauge64;
        description
            "Minimum observed one-way delay.";
    }
    leaf max-delay-value {
        type yang:gauge64;
        description
            "Maximum observed one-way delay.";
    }
    leaf low-delay-percentile {
        type yang:gauge64;
        description
            "Low percentile of observed one-way delay with
             specific measurement method.";
    }
    leaf intermediate-delay-percentile {
        type yang:gauge64;
        description
            "Intermediate percentile of observed one-way delay with
             specific measurement method.";
    }
    leaf high-delay-percentile {
        type yang:gauge64;
        description
            "High percentile of observed one-way delay with
             specific measurement method.";
    }
}

}

grouping link-jitter-statistics {
    description
        "Grouping for per link jitter statistics.";
    container jitter-statistics {
        description
            "One-way link jitter summarized information.";
        reference
            "RFC 3393: IP Packet Delay Variation Metric
             for IP Performance Metrics (IPPM)
             RFC 4656: A One-way Active Measurement Protocol (OWAMP)
             ITU-T Y.1731: Operations, administration and
             maintenance (OAM) functions and mechanisms
             for Ethernet-based networks";
        leaf unit-value {
            type identityref {
                base lime:time-unit-type;
            }
            default "lime:milliseconds";
        }
    }
}
```

```
        description
            "Time units, where the options are s, ms, ns, etc.";
    }
    leaf min-jitter-value {
        type yang:gauge64;
        description
            "Minimum observed one-way jitter.";
    }
    leaf max-jitter-value {
        type yang:gauge64;
        description
            "Maximum observed one-way jitter.";
    }
    leaf low-jitter-percentile {
        type yang:gauge64;
        description
            "Low percentile of observed one-way jitter.";
    }
    leaf intermediate-jitter-percentile {
        type yang:gauge64;
        description
            "Intermediate percentile of observed one-way jitter.";
    }
    leaf high-jitter-percentile {
        type yang:gauge64;
        description
            "High percentile of observed one-way jitter.";
    }
}

grouping tp-svc-telemetry {
    leaf reference-time {
        type yang:date-and-time;
        config false;
        description
            "Indicates the time when the statistics are collected.";
    }
    leaf inbound-octets {
        type yang:counter64;
        description
            "The total number of octets received on the
            interface, including framing characters.";
    }
    leaf inbound-unicast {
        type yang:counter64;
        description
            "The total number of inbound unicast packets.";
```

```
}
leaf inbound-nunicast {
  type yang:counter64;
  description
    "The total number of inbound non-unicast
    (i.e., broadcast or multicast) packets.";
}
leaf inbound-discards {
  type yang:counter64;
  description
    "The number of inbound packets that were chosen to be
    discarded even though no errors had been detected.
    Possible reasons for discarding such a packet could
    be to free up buffer space, not enough buffer for
    too much data, etc.";
}
leaf inbound-errors {
  type yang:counter64;
  description
    "The number of inbound packets that contained errors.";
}
leaf inbound-unknown-protocol {
  type yang:counter64;
  description
    "The number of packets received via the interface
    which were discarded because of an unknown or
    unsupported protocol.";
}
leaf outbound-octets {
  type yang:counter64;
  description
    "The total number of octets transmitted out of the
    interface, including framing characters.";
}
leaf outbound-unicast {
  type yang:counter64;
  description
    "The total number of outbound unicast packets.";
}
leaf outbound-nunicast {
  type yang:counter64;
  description
    "The total number of outbound non unicast
    (i.e., broadcast or multicast) packets.";
}
leaf outbound-discards {
  type yang:counter64;
  description
```

```
        "The number of outbound packets which were chosen
        to be discarded even though no errors had been
        detected to prevent their being transmitted.
        Possible reasons for discarding such a packet could
        be to free up buffer space, not enough buffer for
        too much data, etc.";
    }
    leaf outbound-errors {
        type yang:counter64;
        description
            "The number of outbound packets that contained
            errors.";
    }
    description
        "Grouping for interface service telemetry.";
}

augment "/nw:networks/nw:network/nw:network-types" {
    description
        "Defines the service topologies types.";
    container service-type {
        presence "Indicates network service topology.";
        leaf service-type {
            type identityref {
                base vpn-common:service-type;
            }
            description
                "The presence identifies the network service type,
                e.g., L3VPN, VPLS, etc.";
        }
        description
            "Container for VPN service type.";
    }
}

augment "/nw:networks/nw:network" {
    when 'nw:network-types/nvp:service-type' {
        description
            "Augments only for VPN Network topology.";
    }
    description
        "Augments the network with service topology attributes";
    container vpn-pm-attributes {
        leaf vpn-id {
            type vpn-common:vpn-id;
            description
                "VPN identifier.";
        }
    }
}
```

```
    leaf vpn-service-topology {
      type identityref {
        base vpn-common:vpn-topology;
      }
      description
        "VPN service topology, e.g., hub-spoke, any-to-any,
        hub-spoke-disjoint.";
    }
    description
      "Container for VPN topology attributes.";
  }
}

augment "/nw:networks/nw:network/nw:node" {
  description
    "Augments the network node with other general attributes.";
  container pm-attributes {
    leaf node-type {
      type identityref {
        base node-type;
      }
      description
        "Node type, e.g., PE, P, ASBR.";
    }
    description
      "Container for node attributes.";
    uses entry-summary;
  }
}

augment "/nw:networks/nw:network/nw:node/pm-attributes" {
  when '../nw:network-types/nvp:service-type' {
    description
      "Augments only for VPN node attributes.";
  }
  description
    "Augments the network node with VPN specific attributes.";
  leaf role {
    type identityref {
      base vpn-common:role;
    }
    default "vpn-common:any-to-any-role";
    description
      "Role of the node in the VPN.";
  }
}

augment "/nw:networks/nw:network/nt:link" {
```

```
description
  "Augments the network topology link with performance
   monitoring attributes.";
container pm-attributes {
  description
    "Container for PM attributes.";
  leaf low-percentile {
    type percentile;
    default "10.00";
    description
      "Low percentile to report. Setting low-percentile
       into 0.00 indicates the client is not interested
       in receiving low percentile.";
  }
  leaf intermediate-percentile {
    type percentile;
    default "50.00";
    description
      "Intermediate percentile to report. Setting
       intermediate-percentile into 0.00 indicates the client
       is not interested in receiving intermediate percentile.";
  }
  leaf high-percentile {
    type percentile;
    default "95.00";
    description
      "High percentile to report. Setting high-percentile
       into 0.00 indicates the client is not interested in
       receiving high percentile.";
  }
  leaf measurement-interval {
    type uint32 {
      range "1..max";
    }
    units "seconds";
    default "60";
    description
      "Indicates the time interval to perform PM measurement.";
  }
  leaf start-time {
    type yang:date-and-time;
    config false;
    description
      "The time that the current measurement started.";
  }
  leaf end-time {
    type yang:date-and-time;
    config false;
  }
}
```

```
        description
            "The time that the current measurement ended.";
    }
    leaf pm-source {
        type identityref {
            base pm-source-type;
        }
        config false;
        description
            "The OAM tool used to collect the PM data.";
    }
    container one-way-pm-statistics {
        config false;
        description
            "Container for link telemetry attributes.";
        uses link-loss-statistics;
        uses link-delay-statistics;
        uses link-jitter-statistics;
    }
    list one-way-pm-statistics-per-class {
        key "class-id";
        config false;
        description
            "The list of PM data based on class of service.";
        leaf class-id {
            type string;
            description
                "The class-id is used to identify the class of service.
                This identifier is internal to the administration.";
        }
        uses link-loss-statistics;
        uses link-delay-statistics;
        uses link-jitter-statistics;
    }
}

augment "/nw:networks/nw:network/nt:link/pm-attributes" {
    when '../..//nw:network-types/nvp:service-type' {
        description
            "Augments only for VPN Network topology.";
    }
    description
        "Augments the network topology link with VPN service
        performance monitoring attributes.";
    container vpn-pm-type {
        description
            "The VPN PM type of this logical point-to-point
```

```
    unidirectional VPN link.";
  container inter-vpn-access-interface {
    description
      "Indicates inter-vpn-access-interface PM, which is to
       monitor the performance of logical point-to-point VPN
       connections between a source and a destination
       VPN access interfaces.";
    leaf inter-vpn-access-interface {
      type empty;
      description
        "This is a placeholder for inter-vpn-access-interface PM,
         which is not bound to a specific VPN access interface.
         The source or destination VPN access interface
         of the measurement can be augmented as needed.";
    }
  }
  container underlay-tunnel {
    presence "Enables VPN underlay tunnel PM";
    description
      "Indicates underlay-tunnel PM, which is to monitor
       the performance of underlay tunnels of VPNs.";
    leaf vpn-underlay-transport-type {
      type identityref {
        base vpn-common:protocol-type;
      }
      config false;
      description
        "The leaf indicates the underlay transport type of
         a VPN service, e.g., GRE, LDP, etc.";
    }
  }
}

augment "/nw:networks/nw:network/nw:node/nt:termination-point" {
  description
    "Augments the network topology termination point with
     performance monitoring attributes.";
  container pm-statistics {
    config false;
    description
      "Container for termination point PM attributes.";
    uses tp-svc-telemetry;
  }
}

augment "/nw:networks/nw:network/nw:node"
+ "/nt:termination-point/pm-statistics" {
```



```
when '../.../nw:network-types/nvp:service-type' {
  description
    "Augments only for VPN Network topology.";
}
description
  "Augments the network topology termination-point with
  VPN service performance monitoring attributes";
list vpn-network-access {
  key "network-access-id";
  description
    "The list of PM based on VPN network accesses.";
  leaf network-access-id {
    type vpn-common:vpn-id;
    description
      "References to an identifier for the VPN network
      access, e.g. L3VPN or VPLS.";
  }
  uses tp-svc-telemetry;
}
}
}
}
<CODE ENDS>
```

6. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

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

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). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

- * `"/nw:networks/nw:network/nw:network-types"`: This subtree specifies the VPN service type. Unauthorized access to this subtree may render the VPN service type invalid.
- * `"/nw:networks/nw:network/nvp:vpn-pm-attributes"`: This subtree specifies the VPN service identifier and VPN service topology. Unauthorized access to this subtree may disable the the VPN PM or render the VPN service topology invalid.
- * `"/nw:networks/nw:network/nw:node/nvp:pm-attributes"`: This subtree specifies the network node type and VPN service topology role. Unauthorized access to this subtree may render the node type or VPN service topology invalid.
- * `"/nw:networks/nw:network/nt:link/nvp:pm-attributes"`: This subtree specifies the PM of the network link and VPN link. Unauthorized access to this subtree can impact the network and VPN monitoring.

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. These are the subtrees and data nodes and their sensitivity/vulnerability:

- * `"/nw:networks/nw:network/nw:node/nvp:pm-attributes/nvp:vpn-summary-statistics"`: Unauthorized access to this subtree can disclose the operational state information of VPN instances.
- * `"/nw:networks/nw:network/nt:link/nvp:pm-attributes/nvp:one-way-pm-statistics"`: Unauthorized access to this subtree can disclose the operational state information of network links or VPN abstract links.
- * `"/nw:networks/nw:network/nw:node/nt:termination-point/nvp:pm-statistics"`: Unauthorized access to this subtree can disclose the operational state information of network termination points or VPN network accesses.

7. IANA Considerations

This document requests IANA to register the following URI in the "ns" subregistry within the "IETF XML Registry" [RFC3688]:

URI: `urn:ietf:params:xml:ns:yang:ietf-network-vpn-pm`
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.

This document requests IANA to register the following YANG module in the "YANG Module Names" subregistry [RFC6020] within the "YANG Parameters" registry.

Name: ietf-network-vpn-pm
Namespace: urn:ietf:params:xml:ns:yang:ietf-network-vpn-pm
Maintained by IANA: N
Prefix: nvp
Reference: RFC XXXX (RFC Ed.: replace XXXX with actual RFC number and remove this note.)

8. Acknowledgements

Thanks to Joe Clarke, Adrian Farrel, Tom Petch, Greg Mirsky, Roque Gagliano, Erez Segev, and Dhruv Dhody for reviewing and providing important input to this document.

9. Contributors

The following authors contributed significantly to this document:

Michale Wang
Huawei
Email: wangzitao@huawei.com

Roni Even
Huawei
Email: ron.even.tlv@gmail.com

Change Liu
China Unicom
Email: liuc131@chinaunicom.cn

Honglei Xu
China Telecom
Email: xuhl6@chinatelecom.cn

10. References

10.1. Normative References

[ITU-T-Y-1731]
ITU-T, "Operator Ethernet Service Definition", August 2015, <<https://www.itu.int/rec/T-REC-Y.1731/en>>.

- [RFC3393] Demichelis, C. and P. Chimento, "IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)", RFC 3393, DOI 10.17487/RFC3393, November 2002, <<https://www.rfc-editor.org/info/rfc3393>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <<https://www.rfc-editor.org/info/rfc4364>>.
- [RFC4656] Shalunov, S., Teitelbaum, B., Karp, A., Boote, J., and M. Zekauskas, "A One-way Active Measurement Protocol (OWAMP)", RFC 4656, DOI 10.17487/RFC4656, September 2006, <<https://www.rfc-editor.org/info/rfc4656>>.
- [RFC5357] Hedayat, K., Krzanowski, R., Morton, A., Yum, K., and J. Babiarz, "A Two-Way Active Measurement Protocol (TWAMP)", RFC 5357, DOI 10.17487/RFC5357, October 2008, <<https://www.rfc-editor.org/info/rfc5357>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC6374] Frost, D. and S. Bryant, "Packet Loss and Delay Measurement for MPLS Networks", RFC 6374, DOI 10.17487/RFC6374, September 2011, <<https://www.rfc-editor.org/info/rfc6374>>.
- [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>>.
- [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>>.
- [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>>.
- [RFC8345] Clemm, A., Medved, J., Varga, R., Bahadur, N., Ananthakrishnan, H., and X. Liu, "A YANG Data Model for Network Topologies", RFC 8345, DOI 10.17487/RFC8345, March 2018, <<https://www.rfc-editor.org/info/rfc8345>>.
- [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>>.
- [RFC8532] Kumar, D., Wang, Z., Wu, Q., Ed., Rahman, R., and S. Raghavan, "Generic YANG Data Model for the Management of Operations, Administration, and Maintenance (OAM) Protocols That Use Connectionless Communications", RFC 8532, DOI 10.17487/RFC8532, April 2019, <<https://www.rfc-editor.org/info/rfc8532>>.
- [RFC8571] Ginsberg, L., Ed., Previdi, S., Wu, Q., Tantsura, J., and C. Filsfils, "BGP - Link State (BGP-LS) Advertisement of IGP Traffic Engineering Performance Metric Extensions", RFC 8571, DOI 10.17487/RFC8571, March 2019, <<https://www.rfc-editor.org/info/rfc8571>>.
- [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>>.
- [RFC8762] Mirsky, G., Jun, G., Nydell, H., and R. Foote, "Simple Two-Way Active Measurement Protocol", RFC 8762, DOI 10.17487/RFC8762, March 2020, <<https://www.rfc-editor.org/info/rfc8762>>.

- [RFC9181] Barguil, S., Gonzalez de Dios, O., Ed., Boucadair, M., Ed., and Q. Wu, "A Common YANG Data Model for Layer 2 and Layer 3 VPNs", RFC 9181, DOI 10.17487/RFC9181, February 2022, <<https://www.rfc-editor.org/info/rfc9181>>.

10.2. Informative References

- [I-D.ietf-netmod-node-tags]
Wu, Q., Claise, B., Liu, P., Du, Z., and M. Boucadair, "Data Node Tags in YANG Modules", Work in Progress, Internet-Draft, draft-ietf-netmod-node-tags-07, 29 April 2022, <<https://www.ietf.org/archive/id/draft-ietf-netmod-node-tags-07.txt>>.
- [I-D.ietf-opsawg-l2nm]
Boucadair, M., Dios, O. G. D., Barguil, S., and L. A. Munoz, "A YANG Network Data Model for Layer 2 VPNs", Work in Progress, Internet-Draft, draft-ietf-opsawg-l2nm-15, 29 April 2022, <<https://www.ietf.org/archive/id/draft-ietf-opsawg-l2nm-15.txt>>.
- [I-D.ietf-opsawg-sap]
Boucadair, M., Dios, O. G. D., Barguil, S., Wu, Q., and V. Lopez, "A Network YANG Model for Service Attachment Points (SAPs)", Work in Progress, Internet-Draft, draft-ietf-opsawg-sap-04, 11 April 2022, <<https://www.ietf.org/archive/id/draft-ietf-opsawg-sap-04.txt>>.
- [RFC4026] Andersson, L. and T. Madsen, "Provider Provisioned Virtual Private Network (VPN) Terminology", RFC 4026, DOI 10.17487/RFC4026, March 2005, <<https://www.rfc-editor.org/info/rfc4026>>.
- [RFC5277] Chisholm, S. and H. Trevino, "NETCONF Event Notifications", RFC 5277, DOI 10.17487/RFC5277, July 2008, <<https://www.rfc-editor.org/info/rfc5277>>.
- [RFC7471] Giacalone, S., Ward, D., Drake, J., Atlas, A., and S. Previdi, "OSPF Traffic Engineering (TE) Metric Extensions", RFC 7471, DOI 10.17487/RFC7471, March 2015, <<https://www.rfc-editor.org/info/rfc7471>>.
- [RFC8194] Schoenwaelder, J. and V. Bajpai, "A YANG Data Model for LMAP Measurement Agents", RFC 8194, DOI 10.17487/RFC8194, August 2017, <<https://www.rfc-editor.org/info/rfc8194>>.

- [RFC8309] Wu, Q., Liu, W., and A. Farrel, "Service Models Explained", RFC 8309, DOI 10.17487/RFC8309, January 2018, <<https://www.rfc-editor.org/info/rfc8309>>.
- [RFC8570] Ginsberg, L., Ed., Previdi, S., Ed., Giacalone, S., Ward, D., Drake, J., and Q. Wu, "IS-IS Traffic Engineering (TE) Metric Extensions", RFC 8570, DOI 10.17487/RFC8570, March 2019, <<https://www.rfc-editor.org/info/rfc8570>>.
- [RFC8632] Vallin, S. and M. Bjorklund, "A YANG Data Model for Alarm Management", RFC 8632, DOI 10.17487/RFC8632, September 2019, <<https://www.rfc-editor.org/info/rfc8632>>.
- [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>>.
- [RFC8969] Wu, Q., Ed., Boucadair, M., Ed., Lopez, D., Xie, C., and L. Geng, "A Framework for Automating Service and Network Management with YANG", RFC 8969, DOI 10.17487/RFC8969, January 2021, <<https://www.rfc-editor.org/info/rfc8969>>.
- [RFC9182] Barguil, S., Gonzalez de Dios, O., Ed., Boucadair, M., Ed., Munoz, L., and A. Aguado, "A YANG Network Data Model for Layer 3 VPNs", RFC 9182, DOI 10.17487/RFC9182, February 2022, <<https://www.rfc-editor.org/info/rfc9182>>.

Appendix A. Illustrative Examples

A.1. VPN Performance Subscription Example

The example shown in Figure 8 illustrates how a client subscribes to the performance monitoring information between nodes ('node-id') A and B in the L3 network topology. The performance monitoring parameter that the client is interested in is end-to-end loss.

```
POST /restconf/operations
    /ietf-subscribed-notifications:establish-subscription
{
  "ietf-subscribed-notifications:input":{
    "stream-subtree-filter":{
      "ietf-network:networks":{
        "network":{
          "network-id":"foo:l3-network",
          "ietf-network-vpn-pm:service-type":{
            "ietf-vpn-common:l3vpn":{}
          }
        }
      }
    }
  }
}
```

```

    },
    "node":[
      {
        "node-id":"A",
        "ietf-network-vpn-pm:pm-attributes":{
          "node-type":"PE"
        },
        "termination-point":{
          "tp-id":"1-0-1"
        }
      },
      {
        "node-id":"B",
        "ietf-network-vpn-pm:pm-attributes":{
          "node-type":"PE"
        },
        "termination-point":{
          "tp-id":"2-0-1"
        }
      }
    ],
    "ietf-network-topology:link":{
      "link-id":"A-B",
      "source":{
        "source-node":"A"
      },
      "destination":{
        "dest-node":"B"
      },
      "ietf-network-vpn-pm:pm-attributes":{
        "one-way-pm-statistics":{
          "loss-statistics":{
            "packet-loss-count":{}
          }
        },
        "vpn-underlay-transport-type":"ietf-vpn-common:gre"
      }
    }
  }
},
"ietf-yang-push:periodic":{
  "ietf-yang-push:period":"500"
}
}

```

Figure 8: Pub/Sub Retrieval

A.2. Example of VPN Performance Snapshot

This example, depicted in Figure 9, illustrates an VPN PM instance example in which a client uses RESTCONF [RFC8040] to fetch the performance data of the link and TP belonged to "VPN1".

```

{
  "ietf-network:networks": {
    "network": {
      "network-id": "foo:vpn1",
      "node": [
        {
          "node-id": "A",
          "ietf-network-vpn-pm:pm-attributes": {
            "node-type": "PE"
          },
          "termination-point": {
            "tp-id": "1-0-1",
            "ietf-network-vpn-pm:pm-statistics": {
              "inbound-octets": "100",
              "outbound-octets": "150"
            }
          }
        },
        {
          "node-id": "B",
          "ietf-network-vpn-pm:pm-attributes": {
            "node-type": "PE"
          },
          "termination-point": {
            "tp-id": "2-0-1",
            "ietf-network-vpn-pm:pm-statistics": {
              "inbound-octets": "150",
              "outbound-octets": "100"
            }
          }
        }
      ],
      "ietf-network-topology:link": {
        "link-id": "A-B",
        "source": { "source-node": "A" },
        "destination": { "dest-node": "B" },
        "ietf-network-pm:pm-attributes": {
          "one-way-pm-statistics": {
            "loss-statistics": { "packet-loss-count": "120" }
          },
          "vpn-underlay-transport-type": "ietf-vpn-common:gre"
        }
      }
    }
  }
}

```

Figure 9

A.3. Example of Percentile Monitoring

The following shows an example of a percentile measurement for a VPN link.

```
{
  "ietf-network-topology:link": [
    {
      "link-id": "foo:vpn1-link1",
      "source": {
        "source-node": "vpn-node1"
      },
      "destination": {
        "dest-node": "vpn-node3"
      },
      "ietf-network-vpn-pm:pm-attributes": {
        "low-percentile": "20.00",
        "intermediate-percentile": "50.00",
        "high-percentile": "90.00",
        "one-way-pm-statistics": {
          "delay-statistics": {
            "unit-value": "lime:milliseconds",
            "min-delay-value": "43",
            "max-delay-value": "99",
            "low-delay-percentile": "64",
            "intermediate-delay-percentile": "77",
            "high-delay-percentile": "98"
          }
        },
        "vpn-pm-type": {
          "inter-vpn-access-interface": [null]
        }
      }
    }
  ]
}
```

Authors' Addresses

Bo Wu (editor)
Huawei
101 Software Avenue, Yuhua District
Nanjing
Jiangsu, 210012
China
Email: lana.wubo@huawei.com

Qin Wu (editor)
Huawei
101 Software Avenue, Yuhua District
Nanjing
Jiangsu, 210012
China
Email: bill.wu@huawei.com

Mohamed Boucadair (editor)
Orange
Rennes 35000
France
Email: mohamed.boucadair@orange.com

Oscar Gonzalez de Dios
Telefonica
Madrid
Spain
Email: oscar.gonzalezdedios@telefonica.com

Bin Wen
Comcast
Email: bin_wen@comcast.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: 15 July 2022

D. Li
J. Wu
Tsinghua University
M. Huang
Huawei
L. Qin
Tsinghua University
N. Geng
Huawei
11 January 2022

Source Address Validation: Use Cases and Gap Analysis
draft-li-sav-gap-analysis-01

Abstract

This document identifies the importance and use cases of source address validation (SAV) at both intra-domain level and inter-domain level (see [RFC5210]). Existing intra-domain and inter-domain SAV mechanisms, either Ingress ACL filtering [RFC2827], unicast Reverse Path Forwarding (uRPF) [RFC3704], or Enhanced Feasible-Path uRPF (EFP-uRPF) [RFC8704] has limitations in scalability or accuracy. This document provides gap analysis of the existing SAV mechanisms.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 8174 [RFC8174].

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 15 July 2022.

Copyright Notice

Copyright (c) 2022 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	2
2. Terminology	4
3. Use Cases	4
3.1. Use Case 1: Intra-domain SAV	5
3.2. Use Case 2: Inter-domain SAV	6
4. Gap Analysis	6
4.1. Existing Intra-domain SAV mechanisms	6
4.2. Existing Inter-domain SAV mechanisms	7
5. SAV Requirements	9
6. Security Considerations	9
7. Contributors	9
8. Acknowledgments	9
9. Normative References	9
Authors' Addresses	10

1. Introduction

Source Address Validation (SAV) is important for defending against source address forgery attacks and accurately tracing back to the attackers. Considering that the Internet is extremely large and complex, it is very difficult to solve the source address spoofing problem at a single "level" or through a single SAV mechanism. On the one hand, it is unrealistic to require all networks to deploy a single SAV mechanism. On the other hand, the failure of a single SAV mechanism will completely disable SAV.

To address the issue, Source Address Validation Architecture (SAVA) was proposed [RFC5210]. According to the operating feature of the Internet, SAVA presents a hierarchical architecture which carries out source IP address validation at three checking levels, i.e., access network, intra-domain, and inter-domain. Different levels provide

different granularities of source IP address authenticity. In contrast to the single-level/point model, SAVA allows incremental deployment of SAV mechanisms while keeps effective because of its multiple-fence design. So, enhancing the source IP address validity in all the three checking levels is of high importance. Furthermore, one or more independent and loosely-coupled SAV mechanisms can coexist and cooperate under SAVA, which is friendly to different users (e.g., providers) with different policies or considerations. Obviously, the quality of SAV mechanisms for their target checking levels is key to the performance of SAV.

There are many SAV mechanisms for different checking levels. For the access network level, Source Address Validation Improvement (SAVI) was proposed to force each host to use legitimate source IP address[RFC7039]. SAVI acts as a purely network-based solution without special dependencies on hosts. It dynamically binds each legitimate IP address to a specific port/MAC address and verifies each packet's source address through the binding relationship. One of the most attractive features of SAVI is that it supports the maximally fine granularity of individual IP addresses, which previous ingress filtering mechanisms cannot provide.

At the intra-domain level, static Access Control List (ACL) is a typical solution of SAV. Operators can configure some matching rules to specify which kind of packets are acceptable (or unacceptable). The information of ACL should be updated manually so as to keep consistent with the newest filtering criteria, which inevitably limits the flexibility and accuracy of SAV. Strict unicast Reverse Path Forwarding (uRPF) [RFC3704] is another solution suitable to intra-domain. Routers deploying strict uRPF accept a data packet only when i) the local FIB contains a prefix encompassing the packet's source address and ii) the corresponding forwarding action for the prefix matches the packet's incoming interface. Otherwise, the packet will be dropped. However, in the scenarios (e.g., multihoming cases) where data packets are under asymmetric routing, strict uRPF often improperly blocks legitimate traffic.

At the inter-domain level, a combination of Enhanced Feasible-Path uRPF (EFP-uRPF) and loose uRPF is recommended in[RFC8704]. Particularly, EFP-uRPF is suggested to be applied on customer interfaces. EFP-uRPF on an AS can prevent its customers from spoofing its upstream ASes' source addresses but fails in the case of two customers spoofing each other. On lateral peer interfaces and transit provider interfaces, loose uRPF [RFC3704] is taken. The routers deploying loose uRPF accept any packets whose source addresses appear in the local FIB tables. Due to the loss of directionality, loose uRPF often improperly permits spoofed traffic.

To summarize, given that it is impossible to deploy SAVI on every access network in the Internet, the "fences" at intra- and inter-domain levels are very important for filtering source address forgery packets that are let go by access networks. However, there exist some instinctive drawbacks in the existing SAV mechanisms designed for both the intra- and inter-domain levels, which leads to inevitable improper permit or improper block problems. A more complete SAV mechanism is required for both intra- and inter-domain levels.

This document identifies the use cases of intra- and inter-domain SAVs. These cases will help analyze the instinctive drawbacks of the existing SAV mechanisms. After that, some SAV requirements will be presented.

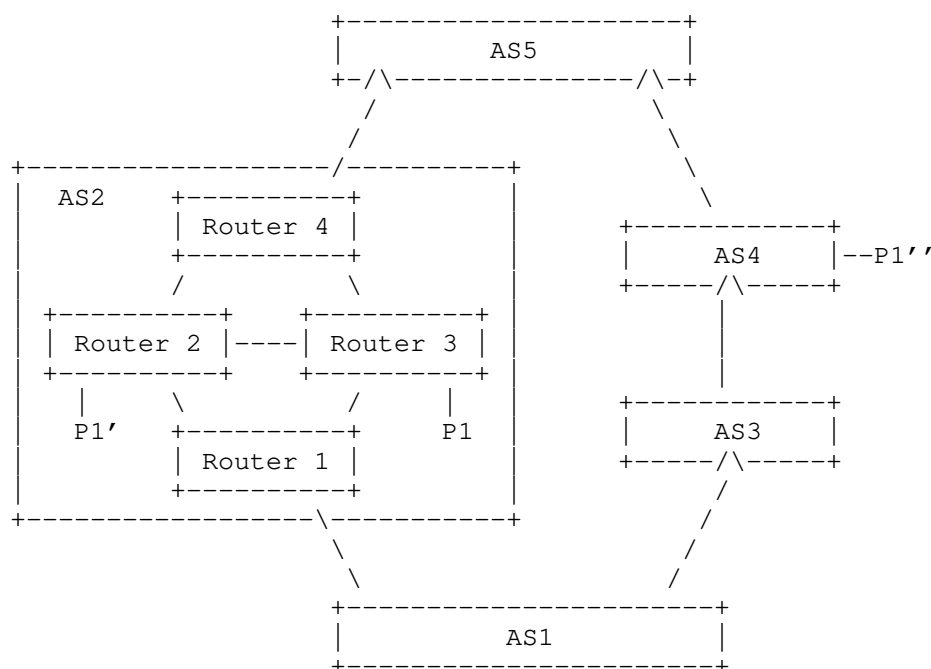
2. Terminology

EAST-WEST traffic denotes the traffic originated and terminated within an AS. Intra-domain SAV aims to check EAST-WEST traffic and prevents hosts/routers from spoofing other source IP blocks in the same AS.

NORTH-SOUTH traffic denotes the traffic arriving from an external AS. Particularly, the traffic arriving from the customer AS is Northward traffic. The traffic received from the provider/peer AS is Southward traffic. Inter-domain SAV aims to verify the authenticity of the source address of NORTH-SOUTH traffic.

3. Use Cases

Figure 1 illustrates the use cases of SAV in both intra- and inter-domain levels. AS1-AS5 belong to the same customer cone, and AS1 is the stub AS. The topology of AS2 is presented while other ASes' inner structures are hidden for brevity.



P1 is the source IP address prefix of Router3.

P1' is the spoofed P1 by Router2 located in the same AS as Router3.

P1'' is the spoofed P1 by Routers located in another AS, i.e., AS4.

Figure 1: Illustration of the use cases of SAV in both intra- and inter-domain levels

3.1. Use Case 1: Intra-domain SAV

In some scenarios especially very large ASes, hosts/routers in the same AS may spoof each other's IP addresses. In Figure 1, Router2 spoofs P1 that originates from Router3. With Intra-domain SAV, EAST-WEST traffic can be checked, and source address spoofing attacks can be prevented. In the figure, Router1, Router3, and Router4 will drop the packets with P1' while accept those with P1, when they deploy Intra-domain SAV mechanisms. Overall, Intra-domain SAV can prevent the source address spoofing from the same AS.

3.2. Use Case 2: Inter-domain SAV

In Figure 1, AS4 spoofs AS2's IP address prefix, i.e., P1 originated from Router3. AS5 will receive the Northward traffic from AS2 and AS4 with legitimate and spoofed IP addresses, respectively. An SAV mechanism is necessary for AS5 to drop the illegal traffic. From the view point of Southward traffic, AS1 may also receive spoofed traffic from AS3 (if AS3 accepts the data packets with source prefix P1"). So, the deployment of SAV on AS1 is also important. Overall, Inter-domain SAV is necessary and can improve the confidence of the source IP address validity among ASes.

4. Gap Analysis

High accuracy is the basic requirement of any intra- or inter-domain SAV mechanism. For any SAV mechanism, improper block problems must be avoided because legitimate traffic must not be influenced. On that basis, SAV should also reduce improper permit problems as much as possible. However, existing SAV mechanisms can not well meet these requirements.

4.1. Existing Intra-domain SAV mechanisms

Operators can configure static ACLs on border routers to validate source addresses. The main drawback of ACL-based SAV is the high operational overhead. Limited application scenarios make the ACL-based method unable to do sufficient SAV on EAST-WEST traffic.

Strict uRPF can generate SAV tables automatically, but it also has limited application scenarios. Figure 2 illustrates an intra-domain scenario. In the scenario, AS1 runs strict uRPF. An access network having IP address prefix 10.0.0.0/15 is attached to two border routers (Router1 and Router2) of AS1. Due to customer's policy, it advertises 10.0.0.0/16 to Router1 and 10.1.0.0/16 to Router2. Then, Router1 and Router2 will advertise the learned IP address prefixes to other routers in AS1 through intra-domain routing protocols such as OSPF and IS-IS.

Although customer only advertises 10.0.0.0/16 to Router1, it may send packets with source IP addresses belonging to 10.1.0.0/16 to Router1 due to load balancing requirements. Suppose the destination node is Router5. Then the path to destination is Customer->Router1->Router3->Router5, while the reverse path is Router5->Router4->Router2->Customer. The round trip routing path is asymmetric, which cannot be dealt with well by strict uRPF.

Specifically speaking, strict uRPF is faced with improper block problems under asymmetric routing scenarios. When Router1/Router3 runs strict uRPF, it learns SAV rules that packets with source address prefix of 10.0.0.0/16 must enter the router on interface '#'. When the packets with source addresses of 10.1.0.0/16 arrive, they will be dropped, which results in improperly blocking legitimate traffic. Similarly, when strict uRPF is deployed on Router2, the improper block problem still exists.

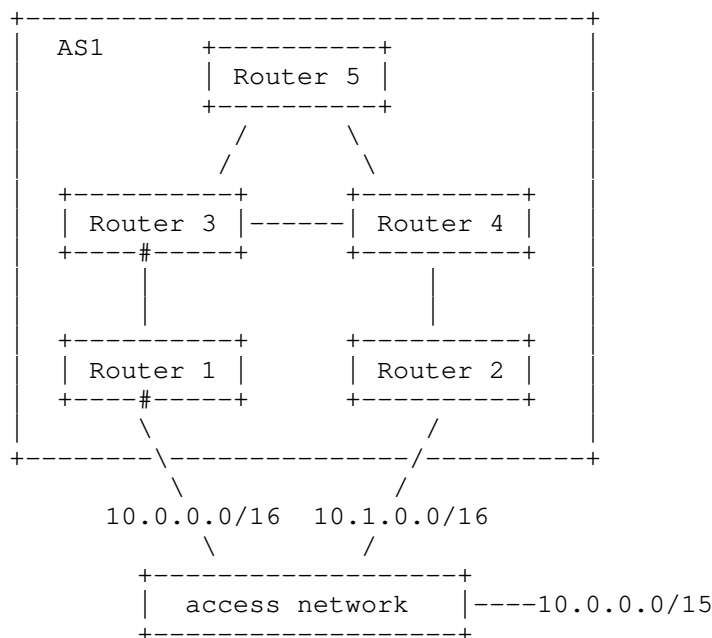


Figure 2: An intra-domain scenario

4.2. Existing Inter-domain SAV mechanisms

The most popular inter-domain SAV is suggested by [RFC8704], which combines EFP-uRPF algorithm B and loose uRPF. In particular, EFP-uRPF algorithm B is for Northward traffic validation. It sacrifices the directionality of customer interfaces for reducing improper permit cases. Loose uRPF is for validating Southward traffic on lateral peer and transit provider interfaces. It sacrifices directionality of Southward traffic completely. Such a combined method sacrificing directionality will leads to improper permit problems sometimes.

Figure 3 illustrates a common inter-domain scenario where the above inter-domain SAV method will fail. In the figure, there are two customer ASes, i.e., AS1 and AS2. Both of them are attached to a

provider AS, i.e., AS4. AS4 has a lateral peer and a provider, i.e., AS3 and AS5. Particularly, AS1 has IP address prefix P1 and advertises it to AS4. IP address prefix P2 is allocated to AS2 and is also advertised to AS4. AS3 has IP address prefix P3 and AS5 has IP address prefix P5. P3 and P5 are also advertised to AS4 through BGP. All arrows represent BGP advertisements. Assume AS4 deploys inter-domain SAV policies, i.e., a combination of EFP-uRPF algorithm B and loose uRPF.

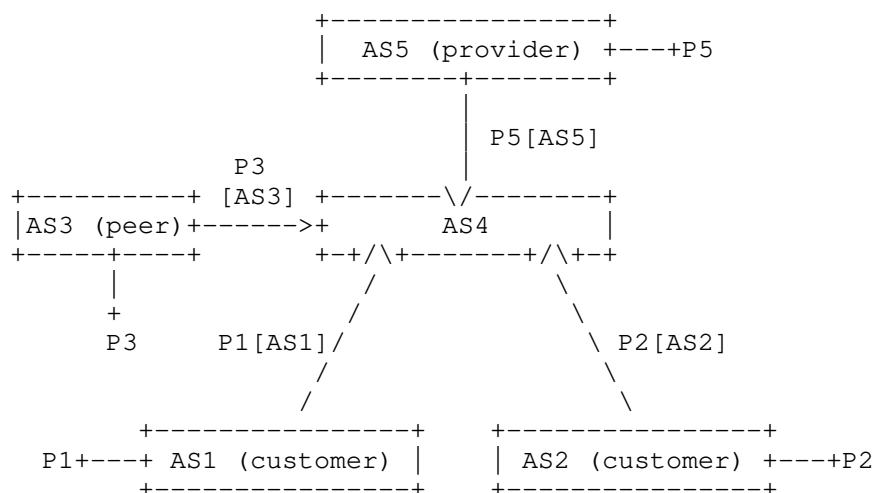


Figure 3: An inter-domain scenario

For Northward traffic, AS4 applies EFP-uRPF. Under EFP-uRPF, AS4 will generate SAV rules considering P1 and P2 are legitimate on both the two customer interfaces. When AS1 spoofs IP address prefix P2 of AS2, the malicious Northward traffic cannot be filtered by AS4. The same is true when AS2 forges P1 of AS1. That is to say, EFP-uRPF cannot prevent source address spoofing among customers even though it only focus on Northward traffic.

For Southward traffic, AS4 deploys loose uRPF for the interfaces of AS3 and AS5. It will learn that the packets with source addresses of P3 or P5 can be accepted without validating the specific arrival interface. Since loose uRPF loses directionality completely, it obviously will fail in dealing with the source address spoofing between its lateral peer and provider, i.e., AS3 and AS5.

5. SAV Requirements

High accuracy, i.e. avoiding improper block problems while trying best to reduce improper permit problems, is the basic requirement of an ideal SAV mechanism. As described above, existing SAV mechanisms cannot meet this requirement. The root cause of their limitations is that they all achieve SAV based on local forwarding information base (FIB) or routing information base (RIB), which may not match the real forwarding direction from the source. In order to guarantee the accuracy, SAV should follow the real data-plane forwarding path. To solve this problem and provide accurate SAV for arbitrary network scenarios, it is required to exchange/explore/probe the forwarding-path information among routers/ASes. In other words, network-wide protocols should be considered.

The network-wide protocols should also consider some practical issues:

- * High scalability. The protocols should not induce much overhead (e.g., bandwidth cost of path probing). Fast convergence under environment changes is also important for improving the scalability in different scales of networks.
- * High deployability. A strategy of incremental deployment needs to be considered. If some routers/ASes do not support the new protocols, improper block should be avoided.
- * High security. The protocols should include mechanisms to guarantee the integrity of protocol packets. Security risks such as Man-in-the-Middle Attack should be avoided.

6. Security Considerations

TBD

7. Contributors

TBD

8. Acknowledgments

TBD

9. Normative References

- [RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, DOI 10.17487/RFC2827, May 2000, <<https://www.rfc-editor.org/info/rfc2827>>.
- [RFC3704] Baker, F. and P. Savola, "Ingress Filtering for Multihomed Networks", BCP 84, RFC 3704, DOI 10.17487/RFC3704, March 2004, <<https://www.rfc-editor.org/info/rfc3704>>.
- [RFC5210] Wu, J., Bi, J., Li, X., Ren, G., Xu, K., and M. Williams, "A Source Address Validation Architecture (SAVA) Testbed and Deployment Experience", RFC 5210, DOI 10.17487/RFC5210, June 2008, <<https://www.rfc-editor.org/info/rfc5210>>.
- [RFC7039] Wu, J., Bi, J., Bagnulo, M., Baker, F., and C. Vogt, Ed., "Source Address Validation Improvement (SAVI) Framework", RFC 7039, DOI 10.17487/RFC7039, October 2013, <<https://www.rfc-editor.org/info/rfc7039>>.
- [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>>.
- [RFC8704] Sriram, K., Montgomery, D., and J. Haas, "Enhanced Feasible-Path Unicast Reverse Path Forwarding", BCP 84, RFC 8704, DOI 10.17487/RFC8704, February 2020, <<https://www.rfc-editor.org/info/rfc8704>>.

Authors' Addresses

Dan Li
Tsinghua University
Beijing
China

Email: tolidan@tsinghua.edu.cn

Jianping Wu
Tsinghua University
Beijing
China

Email: jianping@cernet.edu.cn

Mingqing Huang
Huawei
Beijing
China

Email: huangmingqing@huawei.com

Lancheng Qin
Tsinghua University
Beijing
China

Email: qlc19@mails.tsinghua.edu.cn

Nan Geng
Huawei
Beijing
China

Email: gengnan@huawei.com

OPSA Working Group
Internet-Draft
Intended status: Standards Track
Expires: 23 September 2022

M. Palmero
F. Brockners
Cisco Systems
S. Kumar
NC State University
S. Bhandari
Thoughtspot
C. Cardona
NTT
D. Lopez
Telefonica I+D
22 March 2022

Data Model for Lifecycle Management and Operations
draft-palmero-opsawg-dmlmo-04

Abstract

This document motivates and specifies a data model for lifecycle management and operations. It describes the motivation and requirements to collect asset-centric metrics including but not limited to asset adoption and usability, licensing, supported features and capabilities, enabled features and capabilities, etc.; with the primary objective to measure and improve the overall user experience along the lifecycle journey, from technical requirements and technology selection through advocacy and renewal, including the end of life of an asset.

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 23 September 2022.

Copyright Notice

Copyright (c) 2022 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	3
1.1. Requirements language	4
2. Terminology	4
3. Motivation	5
4. Use Cases	6
4.1. License Inventory and Activation	6
4.2. Features in Use	7
4.3. Assets in Use	8
4.4. Risk Mitigation Check (RMC)	8
4.5. Errata	9
4.6. Security Advisory	9
4.7. Optimal Software Version (OSV)	9
4.7.1. Software Conformance	9
4.7.2. Risk Trend Analysis	10
4.7.3. What-if Analysis	10
4.8. Asset Retirement - End of Life (EOL)	11
5. Information Model	11
6. Data Models	12
6.1. Tree Diagrams of the modules that form LMO	12
6.1.1. Aggregated Asset Inventory	12
6.1.2. Licenses	13
6.1.3. Usage	14
6.1.4. Usage	15
6.1.5. Incident Management	16
6.1.6. Organization	17
6.1.7. User	17
6.2. LMO Modules	18
6.2.1. LMO Common Module	18
7. Deployment Considerations	29
8. Security Considerations	29
9. IANA Considerations	30
9.1. The IETF XML Registry	30

9.2. The YANG Module Names Registry	31
10. References	32
10.1. Normative References	32
10.2. Informative References	32
Acknowledgments	33
Change log	33
Authors' Addresses	35

1. Introduction

The virtualization of hardware assets and the development of applications using microservice architecture for cloud-native infrastructure created new consumption and licensing models. Any service can be deployed by composing multiple assets together where an asset refers to hardware, software, application, system or service. For example, cloud-native infrastructure from one vendor may be hosted on the physical server from another vendor or a combination of multiple cloud-native functions from one or more vendors can be combined to execute any service.

This introduces challenges for both lifecycle and adoption management of the assets. For example, a user may need to identify the capability availability of different assets or measure the usage of each capability (or the combination) from any specific asset to measure its optimal potential. Moreover, the user could pinpoint the reason: the software application could not be optimally deployed, or is not simple to use, or is not well documented, etc. The user may use feed such measurements and analysis metrics back to the support engineers and the developers, so they can focus their work effort only on features that users are adopting, or even determine when the lifecycle of the development could end.

This creates the need to collect and analyze asset-centric lifecycle management and operations data. From now on this data will be referred as Lifecycle Management and Operations (LMO); where LMO is not limited to virtualized or cloud environments, it covers all types of networking environments in which technology assets are deployed.

LMO data constitutes data needed to measure asset-centric lifecycle metrics including but not limited to asset adoption and usability, licensing, supported features and capabilities, enabled features and capabilities, etc. The primary objective is to facilitate the asset lifecycle management from the initial asset selection and positioning, licensing, feature enablement and usage, and beyond renewal to improve the overall user experience.

The main challenge in collecting LMO-related data, especially in a multi-vendor environment, relies on the ability to produce and consume such data in a vendor-agnostic, consistent and synchronized manner. APIs or telemetry are meant to collect and relay this data to receiving equipment for storing, analysis and/or visualization.

This document describes the motivation behind LMO, lists use cases, followed by the information model and data model of LMO. The list of use cases describes the need for new functional blocks and their interactions. The current version of this draft is focused on asset inventory, licenses information, feature usage and incident management. This draft specifies four YANG modules [RFC7950] focused on LMO, including:

- * Licenses,
- * Assets,
- * Usage level of Asset features, and
- * Incident Management.

This document is organized as follows. Section 2 establishes the terminology and abbreviations. In Section 3, the goals and motivation of LMO are discussed. In Section 4, use cases are introduced. Section 5 specifies the information model and the data models for LMO.

1.1. Requirements language

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. Terminology

Terminology and abbreviations used in this document:

- * Asset: refers to hardware, software, applications, or services. An asset can be physical or virtual.
- * Consumer: refers to an entity that utilizes the outcomes of LMO. A consumer can be a user, a developer or some other interested third party.
- * Developer: refers to the entity that creates or develops the entire asset or the part of the asset.
- * EOL: End of Life.

- * **Features:** are options or functional capabilities available in an asset.
- * **License:** is issued by an entity such as the developer or the Open Source community and allows the user to operate the asset. Licenses determine how the asset can be leveraged and what is required in cases the asset is changed.
- * **LMO:** Lifecycle Management and Operations.
- * **Optimal Software Version(OSV):** refers to the elected software version considered optimal in the user environment.
- * **PID:** Product Identifier.
- * **Usage:** refers to how features of the asset are used.
- * **User:** refers to the organization that owns or consumes the asset. Within the organization there are entities that: a) use the assets in their operations, b) manage the assets.
- * **User Experience:** how a user interacts with and experiences a particular asset. It includes a user's perceptions of ease of use, efficiency, and utility of an asset.

3. Motivation

The user experience with a specific asset can be organized into four classes:

1. Asset characteristic class, covering anything related to asset, license, features, etc.
2. Utilization class, to measure how the assets and features are used, duration of usage, uptime, etc.
3. Notification class, covering any security advisory, retirement, etc.
4. Incident class, to record and report any problem the user has faced with the asset.

The ability to measure, produce and consume LMO could benefit the user organization in addressing issues such as:

- * Licenses may not have been obtained at the optimum level for a given feature, where a user might have bought licenses that are not activated.

- * Features of an asset might not be used as needed in all deployments within the organization.
- * Resolution of incidents involving the asset and the developer of the technology used within the asset.

In addition to the resolution of incidents, LMO could allow developer organizations to optimize the features they offer. For example, they could consider deprecating features that are used infrequently or focus on introducing more features for the assets that are widely deployed in various infrastructures.

LMO also covers the need of communication between users and the developer. LMO can provide the capability for users to provide feedback about any asset (e.g., potential deficiency of a feature, feature enhancement request). An administrator in the user organization may include specific metrics that identify a potential problem of that specific feature or a capability of the asset. An engineer in the developer organization can determine the impact of the potential deficiency from the number of users providing feedback. Note that this channel is different from a "call to a Technical Assistance Center" in which the user may request help in resolving operational issues with the asset.

4. Use Cases

4.1. License Inventory and Activation

An operations engineer would like to understand which licenses are activated and which are used and/or consumed. It is also important for asset users to understand which features within their assets might need a license and how to activate them.

It is relatively straightforward to have an inventory of existing licenses when there is only one asset developer (providing the asset) and one asset family.

But complexity grows when there are many different developers, systems and processes involved. New service offerings have introduced new attributes and datasets and require alignment with new business models (pay-per-product, subscription model, pay-as-you-go model, etc.). They might support different license types and models: asset activation keys, trust-based model, systems that act as proxy from the back end owned by the asset developer to support the control of licenses, etc.

Sometimes it is a challenge to report which licenses have been bought by the asset user, or who in the user organization owns that license because that information might rely on different asset developers; even within the same asset developer, licenses may correspond to different types or groups of assets. Asset users often need to interact with different license systems and processes.

Information on how assets are licensed could be delivered from a combination of attributes such as: sales order, purchase order, asset activation key, serial number, etc.

If there is no consistency on how to deal with those data points, complexity increases for the consumer, potentially requiring manual steps. Automating those manual steps or exceptions becomes time-consuming, eventually leading to higher costs for the asset consumer.

Having a common data model for LMO eases the integration between different data sources, processes, and consolidation of the information under a common reference.

4.2. Features in Use

Feature logic is required to identify the configured features from the running configuration and determine how they might be used. There is often a lack of an easy method to list any configured features available in the current asset.

This information is extracted from the running configuration many times, implemented by a rule system without having an easy method to list any configured features available in the current asset.

Some of these use cases need to be built on top of others, and from them, other more complex use cases could be created. For instance, Software Compliance use cases can be automated, based on use cases like security advisory, errata, End of Life(EOL), etc.

All this brings a complete set of use cases that fulfills Lifecycle Management of assets, complementing and providing metrics on how asset users are using assets and how their experience from using those assets can be improved.

4.3. Assets in Use

Current approach to quantify how an asset is used, requires volume or aggregated usage/consumption metrics related to deployed assets, functions, features, integrations, etc. Also the need to quantify which metrics might be associated to a user, an organization, to specific services and how often are used; while others may be based on pre agreed profile (contractual or usage) of intended use.

Examples include:

- * Number of search/queries sent by the user.
- * Amount of data returned to the user.
- * Amount of active time spent using the asset/feature.
- * Number of concurrent users accessing the asset/feature.
- * Number of features in use.
- * Number of users or sites using those features, etc.

The information models and data models for LMO include data fields to support metrics that might be required by consumption-based charging and licensing of asset usage.

4.4. Risk Mitigation Check (RMC)

Network, software and cloud engineers would like to be aware of known issues that are causing assets to crash so that they can act to remediate the issue quickly, or even prevent the crash if alerts are triggered on time. There are analytics tools that can process memory core dumps and crash-related files, providing the ability to the asset developers to determine the root cause.

Accordingly, asset users can remediate the problem, automate the remedy to enable incident deflection, allowing the support staff to focus on new problems. The goal of introducing normalization is not to define attributes for each of the elements being part of the crash information, but the results of RMC should be normalized and registered.

Risk Mitigation Check could also include the possibility to be aware of current and historical restarts allowing network and software engineers to enhance the service quality to asset users.

4.5. Errata

Both hardware and software critical issues or Errata need development to automate asset user matching:

- * Hardware Errata match on product identifiers (PIDs) + serial numbers along with additional hardware attributes.
- * Software Errata match on software type and software version along with some additional device attributes.

Engineering might develop the logic to check whether any critical issue applies to a single serial number or a specific software release.

The information to be correlated includes customer identification, license, and asset information that the asset user might own. All this information needs to be correlated with hardware and software Errata, and EOL information to show which part of the asset inventory might be affected.

4.6. Security Advisory

The Security Advisory use case automates the matching of asset user data to security bulletins published by asset developers. Security Advisory logic implemented by developers could apply to a specific software release.

4.7. Optimal Software Version (OSV)

The objective of the Optimal Software Version (OSV) use case is that consumers can mark software images as OSV for their assets; based on this, it is easier for them to control and align their hardware and software assets to the set of OSVs.

Based on the logic of OSV, use cases like software compliance, risk trend analysis, acknowledge bugs, security advisories, errata, what-if analysis, etc., could be realized.

4.7.1. Software Conformance

All the assets should be at their latest recommended software version in case a security update is required to address a security issue of a specific feature.

The Software Conformance use case provides a view to the asset users and informs the users whether the assets that belong to a specific group conforms to the OSV or not. It can provide the users with a

report, including a representation of software compliance for the entire network and software applications. This report could include the current software version running on the asset and the recommended software version. The report could enable users to quickly highlight which group of assets might need the most attention to inspire appropriate actions.

The Software Conformance use case uses data that might not be provided by the asset itself. Data needs to be provided and maintained also by the asset developers, through e.g., asset catalog information. Similar logic applies to a feature catalog, where the asset developer maintains the data and updates it adequately based on existing bugs, security advisories, etc.

The Software Conformance process needs to correlate the Software catalog information with the software version running on the asset.

4.7.2. Risk Trend Analysis

The Risk Trend Analysis use case provides customers with a risk trend analysis, summarizing what might change before applying changes, including registered bugs, security advisories and errata.

4.7.3. What-if Analysis

The What-if Analysis use case allows asset users to plan for new hardware or software, giving them the possibility to change the config parameters or model how new hardware or software might change the software suggestions generated by OSV.

OSV and the associated use cases involve dependencies on attributes that might need to be collected from assets directly, including related inventory information (serial numbers, asset identifiers, software versions, etc.), but also dynamic information could be required, like:

- * Information on features that might be enabled on the particular asset.
- * Catalogs, that might include information related to release notes. For example, consider a feature catalog. This catalog could include software versions that support a specific feature; the software releases that a feature is supported in; or the latest version that a feature is supported in, in case the feature is EOL.
- * Data sources to correlate information coming from reports on critical issues or errata, security advisory, End of Life, etc.

Those catalogs and data sources with errata information, EOL, etc. need to be maintained and updated by asset developers, making sure, that the software running on the assets is safe to run and up to date.

4.8. Asset Retirement - End of Life (EOL)

Hardware EOL reports need to map Hardware EOL PIDs, focusing on base PIDs so that bundles, spares, non-base PIDs, etc., do not provide false EOL reporting to asset users. Software EOL reports are used to automate the matching of user software type and software version to software EOL bulletins.

5. Information Model

The broad metric classes defined in section 3 that quantify user experience can be modeled as shown in Figure 1. There is an inventory of all assets that the user possesses. Each asset in the inventory may be entitled to one or more licenses; a license may contain one or more sub-licenses. The level of usage for each feature and license associated with the asset is measured. For every asset, a list of incidents could be created.

For example, a user needs to measure the utilization of a specific license for a specific type of asset. The information about the license may reside in a license server. The state (activated or not) of the license may reside with the asset itself or a proxy. They can be aggregated/correlated as per the information model shown in Figure 1 to give information to the user regarding the utilization of the licenses. The user experience is thus enhanced by having accurate knowledge about the utility of the given license.

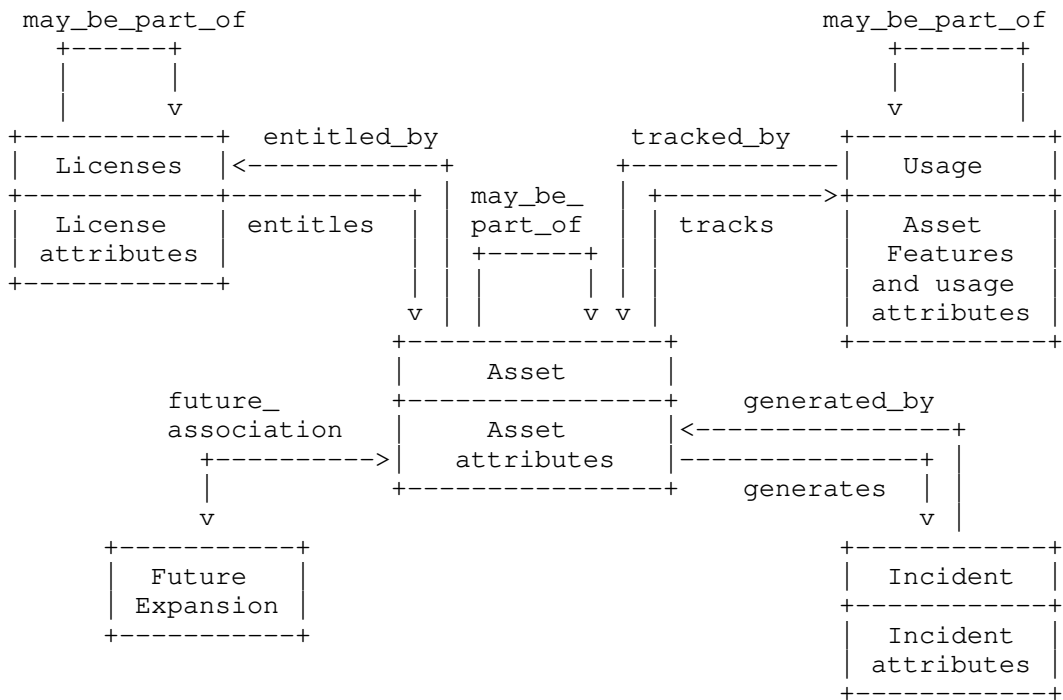


Figure 1: Information Model

The model allows for future expansion by new metrics that will quantify user experience. Notice that future asociation relationship and future expansion might be linked to asset or to one of the other datasets: incident, feature usage or licenses.

6. Data Models

6.1. Tree Diagrams of the modules that form LMO

6.1.1. Aggregated Asset Inventory

This specification uses [I-D.draft-ietf-netmod-geo-location-11], [I-D.draft-ietf-opsawg-sbom-access-03]

module: ietf-lmo-assets-inventory

```

augment /ietf-lmo:lmos/ietf-lmo:lmo/ietf-lmo:inst:
  +--rw vendor?                ietf-lmo-common:vendor-id
  +--rw name?                  string
  +--rw description?           string
  +--rw pid?                   string
  +--rw serial-number?         string
  +--rw vid?                   string
  +--rw mac-address?           yang:mac-address
  +--rw ip-address*            inet:ip-address
  +--rw entity-name?           string
  +--rw product-description?   string
  +--rw udi?                   string
  +--rw transparency-info?     inet:uri
  +--rw role?                  identityref
  +--rw aggregation?           boolean
  +--rw number-of-instances?   uint32
  +--rw platform-dependency-os? identityref
  +--rw install-location
  +--rw deployment-mode?       identityref
  +--rw activation-date?       yang:date-and-time
  +--rw software-version?      string
  +--ro hotfixes
  |   +--ro hostfix* []
  |   |   +--ro version?   identityref
  |   |   +--ro order?     uint8
  +--rw software-type?         string
  +--rw sign-of-life-timestamp? yang:date-and-time
  +--rw tags?                  string

```

6.1.2. Licenses

module: ietf-lmo-licenses

```
augment /ietf-lmo:lmos/ietf-lmo:lmo/ietf-lmo:inst:
  +--rw virtual-account?      string
  +--rw model?                 ietf-lmo-common:license-model-t
  +--rw buying-program?       identityref
  +--rw offer-type?           identityref
  +--rw external-store?       boolean
  +--rw pid?                   string
  +--rw purchase-order-id?     ietf-lmo-common:purchase-order-t
  +--rw account-id?           string
  +--rw assets
  |   +--rw asset* [lmo-class id]
  |   |   +--rw lmo-class      -> /ietf-lmo:lmos/lmo/lmo-class
  |   |   +--rw id             -> deref(..lmo-class)/../ietf-lmo:inst/id
  +--rw features
  |   +--rw feature* [lmo-class id]
  |   |   +--rw lmo-class      -> /ietf-lmo:lmos/lmo/lmo-class
  |   |   +--rw id             -> deref(..lmo-class)/../ietf-lmo:inst/id
  +--rw state?                 ietf-lmo-common:license-state-t
  +--rw renewal-profile
  |   +--rw purchase-date?     yang:date-and-time
  |   +--rw claim-date?       yang:date-and-time
  |   +--rw activation-date?   yang:date-and-time
  |   +--rw expiration-date?   yang:date-and-time
augment /ietf-lmo:lmos/ietf-lmo:lmo/ietf-lmo:inst:
  +--rw licenses
  |   +--rw lmo-class?         -> /ietf-lmo:lmos/lmo/lmo-class
  |   +--rw id?                -> deref(..lmo-class)/../ietf-lmo:inst/id
```

6.1.3. Usage

module: ietf-lmo-feature

```

augment /ietf-lmo:lmos/ietf-lmo:lmo/ietf-lmo:inst:
  +--rw features
    +--rw feature* [lmo-class id]
      +--rw lmo-class    -> /ietf-lmo:lmos/lmo/lmo-class
      +--rw id           -> deref(..lmo-class)/../ietf-lmo:inst/id
augment /ietf-lmo:lmos/ietf-lmo:lmo/ietf-lmo:inst:
  +--rw name?                string
  +--rw summary?             string
  +--rw category?            string
  +--rw entitlement?         string
  +--rw first-available-version? string
  +--ro backported-versions
    | +--ro backported-version* []
    |   +--ro version?  identityref
  +--rw scope?                identityref
  +--rw config-options* [id]
    | +--rw id            string
    | +--rw name?         string
    | +--rw summary?      string
    | +--rw characteristic* [id]
    |   +--rw id          string
    |   +--rw name?       string
    |   +--rw value?      string
  +--rw asset
    | +--rw lmo-class?    -> /ietf-lmo:lmos/lmo/lmo-class
    | +--rw id?           -> deref(..lmo-class)/../ietf-lmo:inst/id
  +--rw subfeatures
    +--rw subfeature* [lmo-class id]
      +--rw lmo-class    -> /ietf-lmo:lmos/lmo/lmo-class
      +--rw id           -> deref(..lmo-class)/../ietf-lmo:inst/id

```

6.1.4. Usage

module: ietf-lmo-usage

```
augment /ietf-lmo:lmos/ietf-lmo:lmo/ietf-lmo:inst:
  +--rw feature
  |   +--rw lmo-class?   -> /ietf-lmo:lmos/lmo/lmo-class
  |   +--rw id?          -> deref(..lmo-class)/../ietf-lmo:inst/id
  +--rw name?            string
  +--rw summary?         string
  +--rw uri?             string
  +--rw deployment-mode? identityref
  +--rw scope?           identityref
  +--rw activation-status? string
  +--rw instances?       uint32
  +--rw count-type?      identityref
  +--rw timestamp?       yang:date-and-time
  +--rw count?           uint32
  +--rw frequency* [name]
  |   +--rw name          string
  |   +--rw type-freq?    string
  |   +--rw value?        yang:counter64
  +--rw resource-consumption* [id]
  |   +--rw id            string
  |   +--rw name?        string
  |   +--rw summary?     string
  |   +--rw characteristic* [id]
  |       +--rw id        string
  |       +--rw name?     string
  |       +--rw unit?     string
  |       +--rw value?    yang:counter64
  |       +--rw value-max? yang:counter64
```

6.1.5. Incident Management

```
module: ietf-lmo-incident-management
```

```
augment /ietf-lmo:lmos/ietf-lmo:lmo/ietf-lmo:inst:
  +--rw id?                string
  +--rw title?              string
  +--rw summary?            string
  +--rw severity?           string
  +--rw status?             string
  +--rw created?            yang:date-and-time
  +--rw last_updated?       yang:date-and-time
  +--rw capability?         string
  +--rw technology?         string
  +--rw subtechnology?      string
  +--rw problem-type?       string
  +--rw resolution?         string
  +--rw owner?              string
  +--rw support-engineer?   string
  +--rw asset
  |   +--rw lmo-class?      -> /ietf-lmo:lmos/lmo/lmo-class
  |   +--rw id?             -> deref(..lmo-class)/../ietf-lmo:inst/id
  +--rw feature
  |   +--rw lmo-class?      -> /ietf-lmo:lmos/lmo/lmo-class
  |   +--rw id?             -> deref(..lmo-class)/../ietf-lmo:inst/id
  +--rw contract-number?    string
```

6.1.6. Organization

```
module: ietf-lmo-organization
```

```
augment /ietf-lmo:lmos/ietf-lmo:lmo/ietf-lmo:inst:
  +--rw address?           string
  +--rw department?        boolean
augment /ietf-lmo:lmos/ietf-lmo:lmo/ietf-lmo:inst:
  +--rw organization
  |   +--rw lmo-class?      -> /ietf-lmo:lmos/lmo/lmo-class
  |   +--rw id?             -> deref(..lmo-class)/../ietf-lmo:inst/id
```

6.1.7. User


```

module: ietf-lmo-user

augment /ietf-lmo:mos/ietf-lmo:lmo/ietf-lmo:inst:
  +--rw billing-account?  uint32
  +--rw represents
  |   +--rw lmo-class?    -> /ietf-lmo:mos/lmo/lmo-class
  |   +--rw id?           -> deref(..lmo-class)/../ietf-lmo:inst/id
  +--rw authority?        enumeration
  +--rw email?            string
augment /ietf-lmo:mos/ietf-lmo:lmo/ietf-lmo:inst:
  +--rw user
  |   +--rw lmo-class?    -> /ietf-lmo:mos/lmo/lmo-class
  |   +--rw id?           -> deref(..lmo-class)/../ietf-lmo:inst/id

```

6.2. LMO Modules

6.2.1. LMO Common Module

```

<CODE BEGINS> file "ietf-lmo-common@2022-02-28.yang"
module ietf-lmo-common {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-lmo-common";
  prefix ietf-lmo-common;
  organization
    "IETF OPSA (Operations and Management Area) Working Group";
  contact
    "WG Web:  <https://datatracker.ietf.org/wg/opsawg/>
    WG List:  <mailto:opsawg@ietf.org>
    Editor:   Marisol Palmero
              <mailto:mpalmero@cisco.com>
    Editor:   Josh Suhr
              <mailto:josuhr@cisco.com>
    Editor:   Sudhendu Kumar
              <mailto:skumar23@ncsu.edu>";
  description
    "This YANG module defines a collection of useful data types
    and identity for Lifecycle Management and Operations (LMO).

    Copyright (c) 2021 IETF Trust and the persons identified as
    authors of the code. All rights reserved.

    Redistribution and use in source and binary forms, with or
    without modification, is permitted pursuant to, and subject to
    the license terms contained in, the Simplified BSD License set
    forth in Section 4.c of the IETF Trust's Legal Provisions

    Relating to IETF Documents
    (https://trustee.ietf.org/license-info)."

```

This version of this YANG module is part of RFC XXXX
(<https://www.rfc-editor.org/info/rfcXXXX>); see the RFC itself
for full legal notices.";

```
revision 2022-02-28 {
  description
    "Introduced flexible root structure";
  reference
    "RFC XXXX: LMO YANG Model";
}
revision 2021-08-23 {
  description
    "Initial revision for Common Module as part of the LMO
    YANG Model";
  reference
    "RFC XXXX: LMO YANG Model";
}

typedef license-id-t {
  type string;
  description
    "License ID Type";
}
typedef license-model-t {
  type enumeration {
    enum perpetual {
      description
        "Perpetual license";
    }
    enum subscription {
      description
        "Subscription license";
    }
    enum usage-based {
      description
        "Usage-based license";
    }
    enum other {
      description
        "Undefined license type";
    }
  }
  description
    "License Model Type";
}
identity license-buying-program-t {
  description
    "License Buying Program that contains the plan to generate
```

```
        revenue for specific asset";
    }
    identity enterprise-agreement {
        base license-buying-program-t;
        description
            "Enterprise Agreement";
    }
    identity managed-service-license-agreement {
        base license-buying-program-t;
        description
            "Managed Service License Agreement";
    }
    identity service-provider-network-agreement {
        base license-buying-program-t;
        description
            "Service Provider Network Agreement";
    }
    identity collab-active-user {
        base license-buying-program-t;
        description
            "Collaboration Active User";
    }
    identity service-full-coverage {
        base license-buying-program-t;
        description
            "Service Full-Coverage";
    }
    identity offer-type-t {
        description
            "License Offer Type, part of the plan to generate revenue
            for specific asset";
    }
    identity perpetual-software {
        base offer-type-t;
        description
            "Perpetual softwar gives the user the right to use the
            program indefinitely";
    }
    identity standalone-hardware {
        base offer-type-t;
        description
            "Standalone hardware is able to function independently
            of other hardware";
    }
    identity on-premise-software-subscription {
        base offer-type-t;
        description
            "On-Premise software subscription, relates to a temporary
```

```
        on-prem licencing model, allowing users to pay a per user
        fee";
    }
    identity cloud-software-saas-subscription {
        base offer-type-t;
        description
            "Cloud Software (SaaS) subscription is a service busines
            model where the user is entitled to use the cloud software
            for a specific time period";
    }
    identity third-party-software {
        base offer-type-t;
        description
            "It includes licenses, agreements, obligations or other
            commitment under which the user can use the asset not
            directly sold by the manufacturer";
    }
    identity flex-cloud-prem-subscription {
        base offer-type-t;
        description
            "Flex Cloud-Prem subscription allows software vendros to
            limit the number of licenses for the use of the specific
            asset";
    }
    typedef license-key-t {
        type string;
        description
            "License Key Type";
    }
    typedef purchase-order-t {
        type string;
        description
            "License purchase order number";
    }
    typedef license-state-t {
        type enumeration {
            enum inactive {
                description
                    "Inactive State";
            }
            enum active {
                description
                    "Active State";
            }
            enum unknown {
                description
                    "Unknown State";
            }
        }
    }
```

```
    }
    description
        "License State Type";
}

typedef asset-id {
    type string;
    description
        "Asset ID Type";
}

typedef vendor-id {
    type enumeration {
        enum cisco {
            description
                "Vendor-id is Cisco";
        }
        enum other {
            description
                "Vendor-id is not determined";
        }
    }
    description
        "Vendor identifier";
}

identity asset-type {
    description
        "type of the asset: hardware, software, software cloud, ...";
}

identity hw {
    base asset-type;
    description
        "Hardware refers to any physical device";
}

identity sw {
    base asset-type;
    description
        "Software refers to a collection of code installed on a
        hardware asset";
}

identity sw-cloud {
    base asset-type;
    description
        "Cloud-based software, that allows users access to software
        application that run on a shared computing resources via
        Internet";
}
```

```
identity phone {
  base asset-type;
  description
    "Mobile telephone or a handheld two-way communication device
    over a cellular network.";
}
identity other {
  base asset-type;
  description
    "Different or additional type not specified as part of another
    defined asset-type.";
}
identity asset-subtype {
  description
    "subtype of the asset: router, switch, wireless,
    controller, ...";
}
identity router {
  base asset-subtype;
  description
    "Network connecting device. It operates at layer-3 of the OSI
    model.";
}
identity switch {
  base asset-subtype;
  description
    "Network connecting device. It operates at layer-2 (Data Link
    Layer) of the OSI model.";
}
identity wireless {
  base asset-subtype;
  description
    "Network connecting device. It creates a wireless local area
    network. It connects to a wired router, switch, or hub via an
    Ethernet cable, and projects a Wi-Fi signal to a designated
    area";
}
identity controller {
  base asset-subtype;
  description
    "Centralized device in the network which is used in combination
    with network connection devices, when there is a need to manage
    them in large quantities.";
}
identity board {
  base asset-subtype;
  description
    "Electronic circuit board in an asset which interconnects
```

```
        another hardware assets attached to it.";
    }
    identity p-supply {
        base asset-subtype;
        description
            "Power supply, as it might have independent identity.";
    }
    identity transceiver {
        base asset-subtype;
        description
            "Device that is both a transmitter and a receiver. Usually
            it's in a single device.
            This is commonly used as a modular network interface";
    }
    identity others {
        base asset-subtype;
        description
            "Different or additional type not specified as part of another
            defined asset-subtype.";
    }
    identity version {
        description
            "Base identity for all version types";
    }
    identity version-sw {
        base version;
        description
            "Version release of the operating system that runs on the
            asset";
    }
    identity platform-dependency-os {
        description
            "Operating system that creates an environment for the asset
            to get deployed. Enum of options covering OS platform
            dependency.";
    }
    identity linux {
        base platform-dependency-os;
        description
            "UNIX like operating system";
    }
    identity windows {
        base platform-dependency-os;
        description
            "Windows operating system";
    }
    identity macOS {
        base platform-dependency-os;
```

```
        description
        "Mac operating system develop by Apple, Inc.";
    }
    identity darwin {
        base platform-dependency-os;
        description
        "Open-source Unix-like operating system first released by Apple
        Inc.";
    }
    identity ubuntu {
        base platform-dependency-os;
        description
        "Linux distribution, used in desktop distribution";
    }
    identity red-hat {
        base platform-dependency-os;
        description
        "Red Hat Enterprise Linux, released in multiple server and
        desktop versions";
    }
    // NEED to extend and include iOS, Android, etc.;

    identity role {
        description
        "What the role of a given device/component is in the network.
        This attribute normally will be configured on the specific
        component during setup. This attribute normally will be
        configured on the specific component during setup";
    }
    identity border-router {
        base role;
        description
        "Router that provides connectivity between interior and
        exterior network routers or to the cloud";
    }
    identity access {
        base role;
        description
        "Router that provides access to a larger communication network
        of some sort.";
    }
    identity control-plane {
        base role;
        description
        "Network data component that controls how data packets are
        forwarded";
    }
    identity edge {
```



```
    base role;
    description
        "Router that provides an entry point into enterprise or service
        provider core networks";
}
identity core {
    base role;
    description
        "Component part of the high-speed backbone of the network. It
        provides fast and efficient data transport";
}
identity datacenter {
    base role;
    description
        "Component placed in the data center, maintaining and housing
        back-end IT system and data stores";
}
identity branch {
    base role;
    description
        "Router in a remote branch of an enterprise's network";
}
identity deployment-mode {
    description
        "This attribute will denote the configured deployment mode
        for the asset and features, if applicable; e.g.,
        High Availability(HA) or Faiover cluster, virtual appliance,
        etc.";
}
identity primary {
    base deployment-mode;
    description
        "Asset or featurts that support critical applications to
        minimize system downtime, to achieve high availabiilty or
        failover";
}
identity secondary {
    base deployment-mode;
    description
        "Redundant asset or feature, that is triggered when the
        primary encounters performance issues, to achieve high
        availability or failover";
}
identity cloud {
    base deployment-mode;
    description
        "Especially it refers to remote, distributed and shared asset
```

```
        resources (i.e. data storage, computing power, etc.), which
        are hooked together and meant to operate as a single
        ecosystem.";
    }
    identity virtual-appliance {
        base deployment-mode;
        description
            "pre-configured virtual machine image, ready to run on a
            hypervisor";
    }
    identity container {
        base deployment-mode;
        description
            "Standard unit of software that packages up code and all its
            dependencies so the application runs quickly and reliably from
            one computing environment to another";
    }
    identity undeployed {
        base deployment-mode;
        description
            "it refers to an asset that is undeployed";
    }
    identity counter-type {
        description
            "Specify the different type of counters, i.e accumulated-count,
            average-count, last-count, high-water mark count, low-water
            mark count" ;
    }
    identity accumulated {
        base counter-type;
        description
            "monotonically increasing counters. They're useful for
            aggregating metric information such as the number of hits
            on a web page, how many users log into a portal, etc.";
    }
    identity average {
        base counter-type;
        description
            "typical value in a set of metrics, in particular the mean,
            which is calculated by dividing the sum of the values in the
            set by their number.";
    }
    identity last {
        base counter-type;
        description
            "Last value measured and collected for specific metric.";
```

```
}
identity high-water-mark {
  base counter-type;
  description
    "Highest level of value in a set of metrics.";
}
identity low-water-mark {
  base counter-type;
  description
    "Lowest level of value in a set of metrics.";
}
identity feature-scope {
  description
    "Optional tag that could apply to any usage feature, so that
     if there are multiple dimensions of reporting that need to
     be accommodated (i.e., report feature usage by 'site')";
}
identity site {
  base feature-scope;
  description
    "Single location, part of the network";
}
identity network {
  base feature-scope;
  description
    "scope limited to the networking assets";
}
typedef feature-usage-type {
  type enumeration {
    enum none {
      description
        "No Usage";
    }
    enum low {
      description
        "Usage meeting the Low Threshold";
    }
    enum medium {
      description
        "Usage meeting the Medium Threshold";
    }
    enum high {
      description
        "Usage meeting the High Threshold";
    }
    // NEED to elaborate more on this list, based on use case
    // validation
  }
}
```

```
    description
      "feature usage % 0-25-50-75-100";
  }

  identity lmo-class {
    description "Base identity for classes of LMOs";
  }
}
<CODE ENDS>
```

7. Deployment Considerations

LMO Data Models defines the data schemas for LMO data. LMO Data Models are based on YANG. YANG data models can be used independent of the transport and can be converted into any encoding format supported by the network configuration protocol. YANG is a protocol independent.

To enable the exchange of LMO data among all interested parties, deployment considerations that are out of the scope of this document, will need to include:

- * The data structure to describe all metrics and quantify relevant data consistently, i.e. specific formats like XML or JSON encoded message would be deemed valid or invalid based on LMO models.
- * The process to share and collect LMO data across the consumers consistently, including the transport mechanism. The LMO YANG models can be used with network management protocols such as NETCONF [RFC6241], RESTCONF [RFC8040], streaming telemetry, etc. OpenAPI specification might also help to consume LMO metrics.
- * How the configuration of assets should be done.

8. Security Considerations

The security considerations mentioned in section 17 of [RFC7950] apply.

LMO brings several security and privacy implications because of the various components and attributes of the information model. For example, each functional component can be tampered with to give manipulated data. LMO when used alone or with other relevant data, can identify an individual, revealing Personal Identifiable Information (PII). Misconfigurations can lead to data being accessed by unauthorized entities.

Methods exist to secure the communication of management information. The transport entity of the functional model MUST implement methods for secure transport. This document also contains an Information model and Data-Model in which none of the objects defined are writable. If the objects are deemed sensitive in a particular environment, access to them MUST be restricted using appropriately configured security and access control rights. The information model contains several optional elements which can be enabled or disabled for the sake of privacy and security. Proper authentication and audit trail MUST be included for all the users/processes that access the LMO.

9. IANA Considerations

9.1. The IETF XML Registry

This document registers URIs in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the registrations defined below are requested:

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

URI: urn:ietf:params:xml:ns:yang:ietf-lmo-common
Registrant Contact: The OPSA WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-lmo-assets-inventory
Registrant Contact: The OPSA WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-lmo-licenses
Registrant Contact: The OPSA WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-lmo-feature
Registrant Contact: The OPSA WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-lmo-usage
Registrant Contact: The OPSA WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-lmo-incident-management
Registrant Contact: The OPSA WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-lmo-organization
Registrant Contact: The OPSA WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-lmo-user
Registrant Contact: The OPSA WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

9.2. The YANG Module Names Registry

This document registers YANG modules in the YANG Module Names registry [RFC7950]. Following the format in [RFC7950], the registrations defined below are requested:

name: ietf-lmo
namespace: urn:ietf:params:xml:ns:yang:ietf-lmo
maintained by IANA: N
prefix: lmocom
reference: RFC XXXX

name: ietf-lmo-common
namespace: urn:ietf:params:xml:ns:yang:ietf-lmo-common
maintained by IANA: N
prefix: lmocom
reference: RFC XXXX

name: ietf-lmo-asset-inventory
namespace: urn:ietf:params:xml:ns:yang:ietf-lmo-assets-inventory
maintained by IANA: N
prefix: lmoasset
reference: RFC XXXX

name: ietf-lmo-licenses
namespace: urn:ietf:params:xml:ns:yang:ietf-lmo-licenses
maintained by IANA: N
prefix: lmolicense
reference: RFC XXXX

name: ietf-lmo-feature
namespace: urn:ietf:params:xml:ns:yang:ietf-lmo-feature
maintained by IANA: N
prefix: lmousage
reference: RFC XXXX

name: ietf-lmo-usage
namespace: urn:ietf:params:xml:ns:yang:ietf-lmo-usage
maintained by IANA: N
prefix: lmousage
reference: RFC XXXX

name: ietf-lmo-incident-management
namespace: urn:ietf:params:xml:ns:yang:ietf-lmo-incident-management
maintained by IANA: N
prefix: lmoscm
reference: RFC XXXX

name: ietf-lmo-organization
namespace: urn:ietf:params:xml:ns:yang:ietf-lmo-organization
maintained by IANA: N
prefix: lmoscm
reference: RFC XXXX

name: ietf-lmo-user
namespace: urn:ietf:params:xml:ns:yang:ietf-lmo-user
maintained by IANA: N
prefix: lmoscm
reference: RFC XXXX

10. References

10.1. 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>>.

10.2. Informative References

- [I-D.draft-ietf-netmod-geo-location-11]
Hopps, C., "A YANG Grouping for Geographic Locations", Work in Progress, Internet-Draft, draft-ietf-netmod-geo-location-11, 11 February 2022, <<https://www.ietf.org/archive/id/draft-ietf-netmod-geo-location-11.txt>>.

- [I-D.draft-ietf-opsawg-sbom-access-03]
Lear, E. and S. Rose, "Discovering and Retrieving Software Transparency and Vulnerability Information", Work in Progress, Internet-Draft, draft-ietf-opsawg-sbom-access-03, 24 October 2021, <<https://www.ietf.org/archive/id/draft-ietf-opsawg-sbom-access-03.txt>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [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>>.

Acknowledgments

The ideas in this document originate from early work by Tony Colon, Carlos Pignataro, and Yenu Gobena originally referred to as Experience Telemetry.

This document was created by meaningful contributions from Josh Suhr, Eric Vyncke, Yannis Viniotis, Nagendra Kumar Nainar, Yenu Gobena, Dhiren Tailor and Jan Lindblad.

The authors wish to thank Gonzalo Salgueiro, Martin Beverley and many others for their helpful comments and suggestions.

Change log

RFC Editor Note: This section is to be removed during the final publication of the document.

version 04

- * Remove ietf-lmo-service YANG module, as service is considered within the asset concept

- * Fix introduced to the .xml and .txt avoiding a compiling issue on the YANG modules.

version 03

- * Flexible root structure has been introduced by the ietf-lmo YANG module: Modules are arranged into layers, with ietf-lmo-common and ietf-lmo at the core. Other modules can be added in layers on top. This structure allows flexibility and the option to be enhanced by vendor implementation. The new structure allows to include other lmo classes, or exclude current lmo classes.
- * Feature and Usage containers have been split in two independent modules. Where Usage relates to runtime data.
- * Organization attribute, has been enhanced to an independent YANG module, adding flexibility and the option to be called independently and enhanced.
- * Service and User YANG modules, have been also introduced in a similar flexible structure, being part of new lmo classes.
- * Information Model, has been enhanced with new modules: Organization, Service and User modules. On this version the new lmo classes can be called independently or from the licenses module. There is no restriction to be called from any of the other YANG modules.

version 02

- * "Support case" renamed to "incident".
- * Add MAC address and IP address attributes under asset-inventory YANG module.
- * Link among objects & YANG modules (notably with feature).
- * New text about asset usage.

version 01

- * Fixes for YANG validator and idnits warnings.

version 00

- * Initial version.

Authors' Addresses

Marisol Palmero
Cisco Systems
Email: mpalmero@cisco.com

Frank Brockners
Cisco Systems
Email: fbrockne@cisco.com

Sudhendu Kumar
NC State University
Email: skumar23@ncsu.edu

Shwetha Bhandari
Thoughtspot
Email: shwetha.bhandari@thoughtspot.com

Camilo Cardona
NTT
Email: camilo@ntt.net

Diego Lopez
Telefonica I+D
Email: diego.r.lopez@telefonica.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: 7 April 2022

M. Tuexen, Ed.
Muenster Univ. of Appl. Sciences
F. Rizzo
Politecnico di Torino
J. Bongertz
Airbus DS CyberSecurity
G. Combs
Wireshark
G. Harris

E. Chaudron
Red Hat
M. Richardson
Sandelman
4 October 2021

PCAP Next Generation (pcapng) Capture File Format
draft-tuexen-opsawg-pcapng-04

Abstract

This document describes a format to record captured packets to a file. This format is extensible; Wireshark can currently read and write it, and libpcap can currently read some pcapng files.

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 7 April 2022.

Copyright Notice

Copyright (c) 2021 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 Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	3
2.1. Acronyms	3
3. General File Structure	4
3.1. General Block Structure	4
3.2. Block Types	5
3.3. Logical Block Hierarchy	6
3.4. Physical File Layout	7
3.5. Options	8
3.5.1. Custom Options	12
3.6. Data format	13
3.6.1. Endianness	13
3.6.2. Alignment	14
4. Block Definition	14
4.1. Section Header Block	14
4.2. Interface Description Block	18
4.3. Enhanced Packet Block	25
4.3.1. Enhanced Packet Block Flags Word	30
4.4. Simple Packet Block	31
4.5. Name Resolution Block	33
4.6. Interface Statistics Block	37
4.7. Decryption Secrets Block	40
4.8. Custom Block	44
5. Vendor-Specific Custom Extensions	46
5.1. Supported Use-Cases	46
5.2. Controlling Copy Behavior	47
5.3. Strings vs. Octets	47
5.4. Endianness Issues	47
6. Recommended File Name Extension: .pcapng	48
7. Conclusions	49
8. Implementations	49
9. Security Considerations	49
10. IANA Considerations	49
10.1. Standardized Block Type Codes	50
11. Contributors	53
12. Acknowledgments	53
13. References	53

13.1. Normative References	53
13.2. Informative References	53
Appendix A. Packet Block (obsolete!)	53
Authors' Addresses	56

1. Introduction

The problem of exchanging packet traces becomes more and more critical every day; unfortunately, no standard solutions exist for this task right now. One of the most accepted packet interchange formats is the one defined by libpcap, which is rather old and is lacking in functionality for more modern applications particularly from the extensibility point of view.

This document proposes a new format for recording packet traces. The following goals are being pursued:

Extensibility: It should be possible to add new standard capabilities to the file format over time, and third parties should be able to enrich the information embedded in the file with proprietary extensions, with tools unaware of newer extensions being able to ignore them.

Portability: A capture trace must contain all the information needed to read data independently from network, hardware and operating system of the machine that made the capture.

Merge/Append data: It should be possible to add data at the end of a given file, and the resulting file must still be readable.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2.1. Acronyms

The following acronyms are used throughout this document:

SHB: Section Header Block

IDB: Interface Description Block

ISB: Interface Statistics Block

EPB: Enhanced Packet Block

SPB: Simple Packet Block

NRB: Name Resolution Block

CB: Custom Block

3. General File Structure

3.1. General Block Structure

A capture file is organized in blocks, that are appended one to another to form the file. All the blocks share a common format, which is shown in Figure 1.

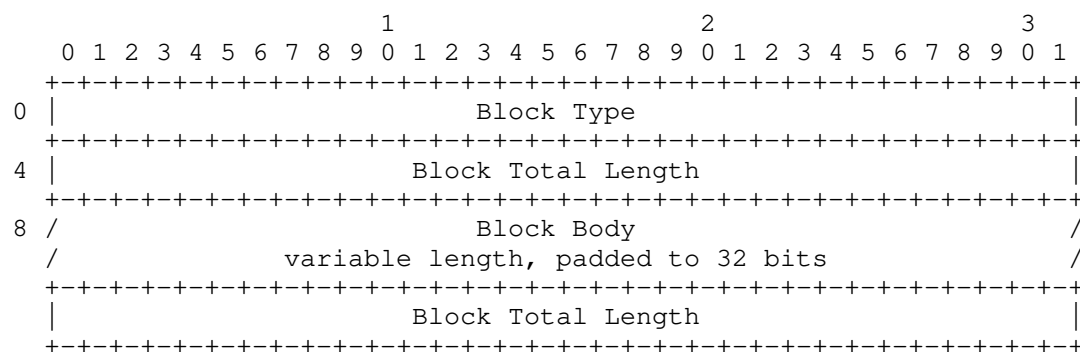


Figure 1: Basic block structure.

The fields have the following meaning:

- * Block Type (32 bits): a unique unsigned value that identifies the block. Values whose Most Significant Bit (MSB) is equal to 1 are reserved for local use. They can be used to make extensions to the file format to save private data to the file. The list of currently defined types can be found in Section 10.1.
- * Block Total Length (32 bits): an unsigned value giving the total size of this block, in octets. For instance, the length of a block that does not have a body is 12 octets: 4 octets for the Block Type, 4 octets for the initial Block Total Length and 4 octets for the trailing Block Total Length. This value MUST be a multiple of 4.
- * Block Body: content of the block.

- * Block Total Length: total size of this block, in octets. This field is duplicated to permit backward file navigation.

This structure, shared among all blocks, makes it easy to process a file and to skip unneeded or unknown blocks. Some blocks can contain other blocks inside (nested blocks). Some of the blocks are mandatory, i.e. a capture file is not valid if they are not present, other are optional.

The General Block Structure allows defining other blocks if needed. A parser that does not understand them can simply ignore their content.

3.2. Block Types

The currently standardized Block Type codes are specified in Section 10.1; they have been grouped in the following four categories:

The following MANDATORY block MUST appear at least once in each file:

- * Section Header Block (Section 4.1): it defines the most important characteristics of the capture file.

The following OPTIONAL blocks MAY appear in a file:

- * Interface Description Block (Section 4.2): it defines the most important characteristics of the interface(s) used for capturing traffic. This block is required in certain cases, as described later.
- * Enhanced Packet Block (Section 4.3): it contains a single captured packet, or a portion of it. It represents an evolution of the original, now obsolete, Packet Block (Appendix A). If this appears in a file, an Interface Description Block is also required, before this block.
- * Simple Packet Block (Section 4.4): it contains a single captured packet, or a portion of it, with only a minimal set of information about it. If this appears in a file, an Interface Description Block is also required, before this block.
- * Name Resolution Block (Section 4.5): it defines the mapping from numeric addresses present in the packet capture and the canonical name counterpart.

- * Interface Statistics Block (Section 4.6): it defines how to store some statistical data (e.g. packet dropped, etc) which can be useful to understand the conditions in which the capture has been made. If this appears in a file, an Interface Description Block is also required, before this block.
- * Custom Block (Section 4.8): it contains vendor-specific data in a portable fashion.

The following OBSOLETE block SHOULD NOT appear in newly written files (but is documented in the Appendix for reference):

- * Packet Block (Appendix A): it contains a single captured packet, or a portion of it. It is OBSOLETE, and superseded by the Enhanced Packet Block (Section 4.3).

The following EXPERIMENTAL blocks are considered interesting but the authors believe that they deserve more in-depth discussion before being defined:

- * Alternative Packet Blocks
- * Compression Block
- * Encryption Block
- * Fixed Length Block
- * Directory Block
- * Traffic Statistics and Monitoring Blocks
- * Event/Security Blocks

Requests for new standardized Block Type codes should be made by creating a pull request to update this document as described in Section 10.1.

3.3. Logical Block Hierarchy

The blocks build a logical hierarchy as they refer to each other. Figure 2 shows the logical hierarchy of the currently defined blocks in the form of a "tree view":


```

Section Header
|
+-- Interface Description
|   +- Simple Packet
|   +- Enhanced Packet
|   +- Interface Statistics
|
+-- Name Resolution

```

Figure 2: Logical Block Hierarchy of a pcapng File

For example: each captured packet refers to a specific capture interface, the interface itself refers to a specific section.

3.4. Physical File Layout

The file MUST begin with a Section Header Block. However, more than one Section Header Block can be present in the capture file, each one covering the data following it until the next one (or the end of file). A Section includes the data delimited by two Section Header Blocks (or by a Section Header Block and the end of the file), including the first Section Header Block.

In case an application cannot read a Section because of different version number, it MUST skip everything until the next Section Header Block. Note that, in order to properly skip the blocks until the next section, all blocks MUST have the fields Type and Length at the beginning. In order to properly skip blocks in the backward direction, all blocks MUST have the Length repeated at the end of the block. These are mandatory requirements that MUST be maintained in future versions of the block format.

Figure 3 shows a typical file layout, with a single Section Header that covers the whole file.

```

+-----+
| SHB v1.0 |                               Data                               |
+-----+

```

Figure 3: File structure example: Typical layout with a single Section Header Block

Figure 4 shows a file that contains three headers, and is normally the result of file concatenation. An application that understands only version 1.0 of the file format skips the intermediate section and restart processing the packets after the third Section Header.

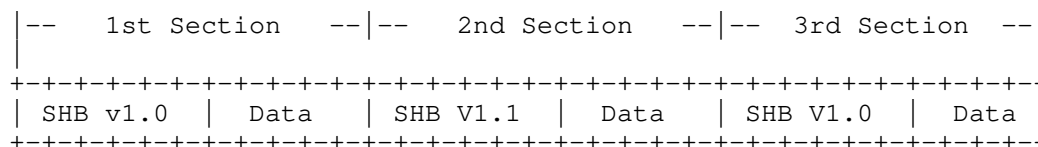


Figure 4: File structure example: three Section Header Blocks in a single file

Figure 5 shows a file comparable to a "classic libpcap" file - the minimum for a useful capture file. It contains a single Section Header Block (SHB), a single Interface Description Block (IDB) and a few Enhanced Packet Blocks (EPB).

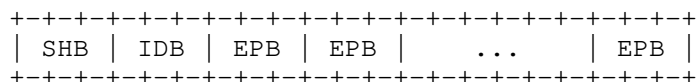


Figure 5: File structure example: a pcapng file similar to a classical libpcap file

Figure 6 shows a complex example file. In addition to the minimum file above, it contains packets captured from three interfaces, capturing on the third of which begins after packets have arrived on other interfaces, and also includes some Name Resolution Blocks (NRB) and an Interface Statistics Block (ISB).

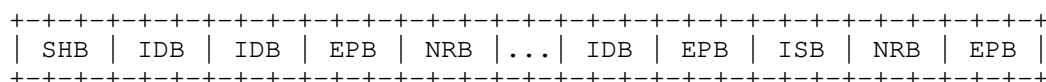


Figure 6: File structure example: complex pcapng file

The last example should make it obvious that the block structure makes the file format very flexible compared to the classical libpcap format.

3.5. Options

All the block bodies MAY embed optional fields. Optional fields can be used to insert some information that may be useful when reading data, but that is not really needed for packet processing. Therefore, each tool can either read the content of the optional fields (if any), or skip some of them or even all at once.

A block that may contain options must be structured so that the number of octets of data in the Block Body that precede the options can be determined from that data; that allows the beginning of the

options to be found. That is true for all standard blocks that support options; for Custom Blocks that support options, the Custom Data must be structured in such a fashion. This means that the Block Length field (present in the General Block Structure, see Section 3.1) can be used to determine how many octets of optional fields, if any, are present in the block. That number can be used to determine whether the block has optional fields (if it is zero, there are no optional fields), to check, when processing optional fields, whether any optional fields remain, and to skip all the optional fields at once.

Options are a list of Type - Length - Value fields, each one containing a single value:

- * Option Type (16 bits): an unsigned value that contains the code that specifies the type of the current TLV record. Option types whose Most Significant Bit is equal to one are reserved for local use; therefore, there is no guarantee that the code used is unique among all capture files (generated by other applications), and is most certainly not portable. For cross-platform globally unique vendor-specific extensions, the Custom Option MUST be used instead, as defined in Section 3.5.1).
- * Option Length (16 bits): an unsigned value that contains the actual length of the following 'Option Value' field without the padding octets.
- * Option Value (variable length): the value of the given option, padded to a 32-bit boundary. The actual length of this field (i.e. without the padding octets) is specified by the Option Length field.

Requests for new standardized option codes for a given block should be made by creating a pull request to update this document as described in Section 10.1.

A given option may have a fixed length, in which case all instances of that option have a length that is equal to the specified fixed length, or a variable length, in which case the option has a minimum length and all instances of that option must have a length equal to or greater than the specified minimum length. The length of fixed-length options, and the minimum length of variable-length options, is specified in the description of the option; if the minimum length of a variable-length option is not specified, a zero-length option is valid. Software that reads these files SHOULD report options that have an invalid length as errors; the software MAY stop processing the file if it sees an option that has invalid length, or MAY ignore the option and continue processing it. Software that writes these files MUST NOT write files with options that have invalid lengths.

If an option's value is a string, the value is not necessarily zero-terminated. Software that reads these files MUST NOT assume that strings are zero-terminated, and MUST treat a zero-value octet as a string terminator.

Some options may be repeated several times; for example, a block can have multiple comments, and an Interface Description Block can give multiple IPv4 or IPv6 addresses for the interface if it has multiple IPv4 or IPv6 addresses assigned to it. Other options may appear at most once in a given block.

The option list is terminated by a option which uses the special 'End of Option' code (opt_endofopt). Code that writes pcapng files MUST put an opt_endofopt option at the end of an option list. Code that reads pcapng files MUST NOT assume an option list will have an opt_endofopt option at the end; it MUST also check for the end of the block, and SHOULD treat blocks where the option list has no opt_endofopt option as if the option list had an opt_endofopt option at the end.

The format of the optional fields is shown in Figure 7.

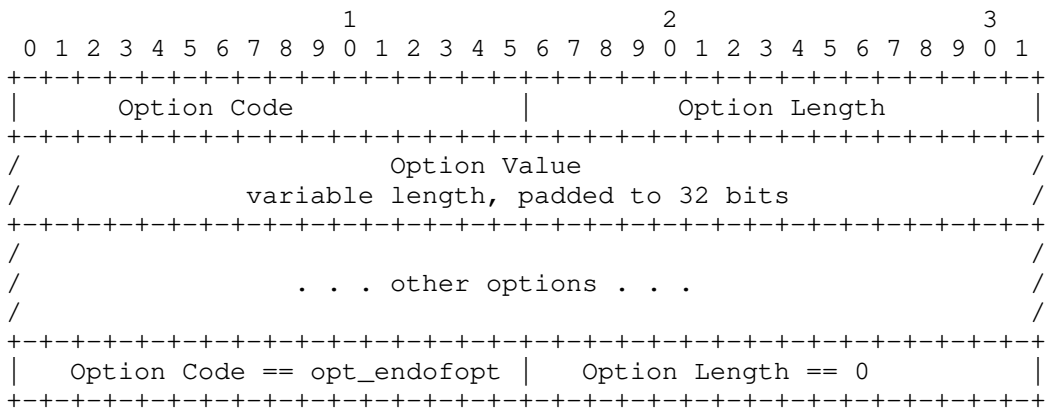


Figure 7: Options Format

The following codes can always be present in any optional field:

Name	Code	Length	Multiple allowed?
opt_endofopt	0	0	no
opt_comment	1	variable	yes
opt_custom	2988/2989/19372/19373	variable, minimum 4	yes

Table 1: Common Options

- opt_endofopt:
The opt_endofopt option delimits the end of the optional fields. This option MUST NOT be repeated within a given list of options.
- opt_comment:
The opt_comment option is a UTF-8 string containing human-readable comment text that is associated to the current block. Line separators SHOULD be a carriage-return + linefeed ('\r\n') or just linefeed ('\n'); either form may appear and be considered a line separator. The string is not zero-terminated.

Examples: "This packet is the beginning of all of our problems",
 "Packets 17-23 showing a bogus TCP retransmission!\r\n This is
 reported in bugzilla entry 1486.\nIt will be fixed in the future.".

opt_custom:

This option is described in detail in Section 3.5.1.

3.5.1. Custom Options

Customs Options are used for portable, vendor-specific data related to the block they're in. A Custom Option can be in any block type that can have options, can be repeated any number of times in a block, and may come before or after other option types - except the opt_endofopt option, which is always the last option. Different Custom Options, of different type codes and/or different Private Enterprise Numbers, may be used in the same pcapng file. See Section 5 for additional details.

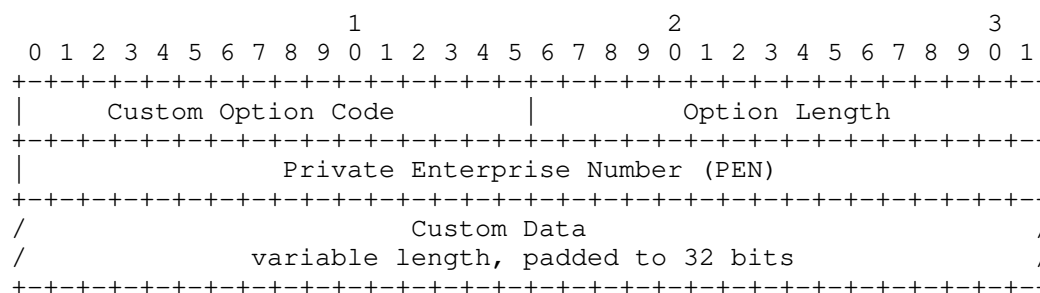


Figure 8: Custom Options Format

The Custom Option has the following fields:

- * Custom Option Code: The code number for the Custom Option, which can be one of the following decimal numbers:

2988:

This option code identifies a Custom Option containing a UTF-8 string in the Custom Data portion. The string is not zero-terminated. This Custom Option can be safely copied to a new file if the pcapng file is manipulated by an application; otherwise 19372 should be used instead. See Section 5.2 for details.

2989:

This option code identifies a Custom Option containing binary octets in the Custom Data portion. This Custom Option can be safely copied to a new file if the pcapng file is manipulated by an application; otherwise 19372 should be used instead. See Section 5.2 for details.

19372:

This option code identifies a Custom Option containing a UTF-8 string in the Custom Data portion. The string is not zero-terminated. This Custom Option should not be copied to a new file if the pcapng file is manipulated by an application. See Section 5.2 for details.

19373:

This option code identifies a Custom Option containing binary octets in the Custom Data portion. This Custom Option should not be copied to a new file if the pcapng file is manipulated by an application. See Section 5.2 for details.

- * Option Length: as described in Section 3.1, this contains the length of the option's value, which includes the 4-octet Private Enterprise Number and variable-length Custom Data fields, without the padding octets.
- * Private Enterprise Number: An IANA-assigned Private Enterprise Number identifying the organization which defined the Custom Option. See Section 5.1 for details. The PEN MUST be encoded using the same endianness as the Section Header Block it is within the scope of.
- * Custom Data: the custom data, padded to a 32 bit boundary.

3.6. Data format

3.6.1. Endianness

Data contained in each section will always be saved according to the characteristics (little endian / big endian) of the capturing machine. This refers to all the fields that are saved as numbers and that span over two or more octets.

The approach of having each section saved in the native format of the generating host is more efficient because it avoids translation of data when reading / writing on the host itself, which is the most common case when generating/processing capture captures.

Please note: The endianness is indicated by the Section Header Block (Section 4.1). Since this block can appear several times in a pcapng file, a single file can contain both endianness variants.

3.6.2. Alignment

All fields of this specification use proper alignment for 16- and 32-bit values. This makes it easier and faster to read/write file contents if using techniques like memory mapped files.

The alignment octets (marked in this document e.g. with "padded to 32 bits") MUST be filled with zeroes.

Please note: 64-bit values are not aligned to 64-bit boundaries. This is because the file is naturally aligned to 32-bit boundaries only. Special care MUST be taken when reading and writing such values. (Note also that some 64-bit values are represented as a 64-bit integer in the endianness of the machine that wrote the file, and others are represented as 2 32-bit values, one containing the upper 32 bits of the value and one containing the lower 32 bits of the value, each written as 32-bit integers in the endianness of the machine that wrote the file. Neither of these formats guarantee 64-bit alignment.)

4. Block Definition

This section details the format of the blocks currently defined.

4.1. Section Header Block

The Section Header Block (SHB) is mandatory. It identifies the beginning of a section of the capture file. The Section Header Block does not contain data but it rather identifies a list of blocks (interfaces, packets) that are logically correlated. Its format is shown in Figure 9.

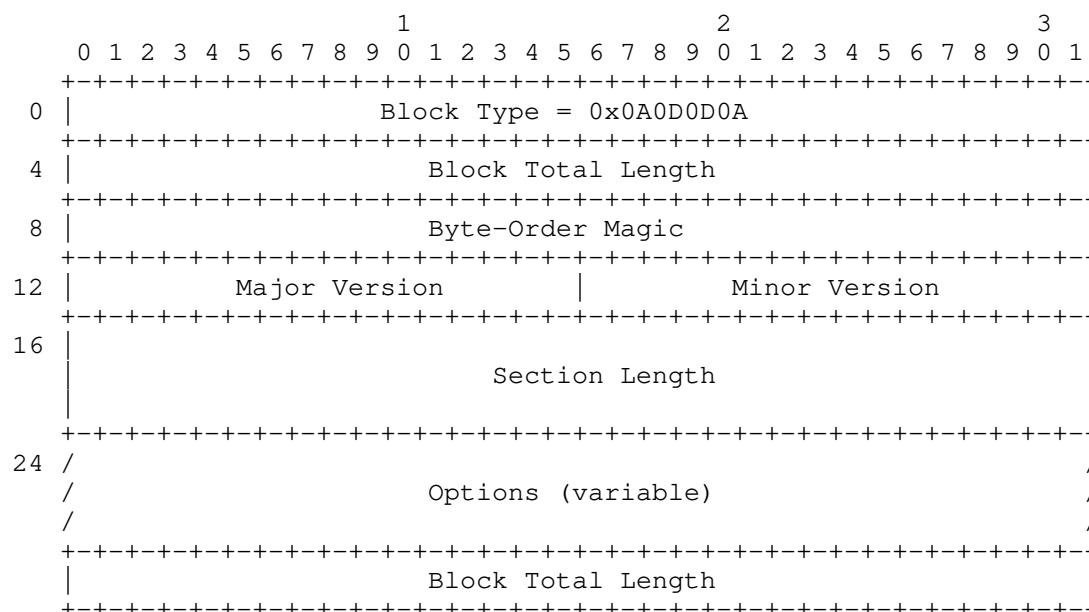


Figure 9: Section Header Block Format

The meaning of the fields is:

- * **Block Type:** The block type of the Section Header Block is the integer corresponding to the 4-char string "\n\r\r\n" (0x0A0D0D0A). This particular value is used for 2 reasons:
 1. This number is used to detect if a file has been transferred via FTP or HTTP from a machine to another with an inappropriate ASCII conversion. In this case, the value of this field will differ from the standard one ("\n\r\r\n") and the reader can detect a possibly corrupted file.
 2. This value is palindromic, so that the reader is able to recognize the Section Header Block regardless of the endianness of the section. The endianness is recognized by reading the Byte Order Magic, which is located 8 octets after the Block Type.
- * **Block Total Length:** total size of this block, as described in Section 3.1.

- * **Byte-Order Magic (32 bits):** an unsigned magic number, whose value is the hexadecimal number 0x1A2B3C4D. This number can be used to distinguish sections that have been saved on little-endian machines from the ones saved on big-endian machines, and to heuristically identify pcapng files.
- * **Major Version (16 bits):** an unsigned value, giving the number of the current major version of the format. The value for the current version of the format is 1.
- * **Minor Version (16 bits):** an unsigned value, giving the number of the current minor version of the format. The value for the current version of the format is 0.
- * **Section Length (64 bits):** a signed value specifying the length in octets of the following section, excluding the Section Header Block itself. This field can be used to skip the section, for faster navigation inside large files. If the Section Length is -1 (0xFFFFFFFFFFFFFFFF), this means that the size of the section is not specified, and the only way to skip the section is to parse the blocks that it contains. Please note that if this field is valid (i.e. not negative), its value is always a multiple of 4, as all the blocks are aligned to and padded to 32-bit (4 octet) boundaries. Also, special care should be taken in accessing this field: since the alignment of all the blocks in the file is 32-bits, this field is not guaranteed to be aligned to a 64-bit boundary. This could be a problem on 64-bit processors.
- * **Options:** optionally, a list of options (formatted according to the rules defined in Section 3.5) can be present.

Writers of pcapng files **MUST NOT** write SHBs with a Major Version other than 1 or a Minor Version other than 0. If they do so, they will write a file that many readers of pcapng files, such as programs using libpcap to read pcapng files (including, but not limited to, tcpdump), Wireshark, and possibly other programs not to be able to read their files.

Some pcapng file writers have used a minor version of 2, but the file format did not change incompatibly (new block types were added); Readers of pcapng files **MUST** treat a Minor Version of 2 as equivalent to a Minor Version of 0 (and, if they also write a pcapng file based on the results of reading one or more pcapng files, they **MUST NOT**, as per the previous sentence, write an SHB with a Minor Version of 2, even if they read an SHB with a Minor Version of 2). As indicated above, using a minor version number other than 0 when writing a section of a pcapng file will produce a section that most existing software will not be able to read; future versions of some of that

software will be able to read sections with a version of 1.2, but older copies of that software that are not updated to the latest version will still not be able to read them.

The Major Version would be changed only if a new version of this specification, for a later major version of the file format, were created. Such a version would only be created if the format were to change in such a way that code that reads the new format could not read the old format (i.e., code to read both formats would have to check the version number and use different code paths for the two formats) and code that reads the old format could not read the new format. An incompatible change to the format of an existing block or an existing option would be such a change; the addition of a new block or a new option would not be such a change. An example of such an incompatible change would be the addition of an additional field to the Section Header Block, following the Minor Version field and before the Snaplen field; software expecting the new SHB format would not correctly read the old SHB format, and software expecting the old SHB format would not correctly read the new SHB format. (Note that a change to the SHB must leave the Block Type, Block Total Length, Byte-Order Magic, Major Version, and Minor Version fields at the same offsets from the beginning of the SHB and with the same lengths, must keep the value of the Block Type the same, must keep the two possible values of the Byte-Order Magic the same, depending on the block's byte order, so that the rest of the SHB can be correctly interpreted.)

The Minor Version would be changed only if a new version of this specification, for a later minor version of the file format, were created. Such a version would only be created if the format were to change in such a way that code that reads the new format could read the old format without checking the version number but code that reads the old format could not read all files in the new format. A backward-compatible change to the format of an existing block or an existing option would be such a change; the addition of a new block or a new option would not be such a change. An example of such a backward-compatible but not forward-compatible change would be a change to the Interface Description block (see below) to use the current Reserved field to indicate the presence of additional fields before the Options, with a zero value indicate no such fields are present.

I.e., adding new block types or options would not require that either the Major Version or the Minor Version be changed, as code that does not know about the block type or option should just skip it; only if skipping a block or option does not work should the minor version number be changed.

Aside from the options defined in Section 3.5, the following options are valid within this block:

Name	Code	Length	Multiple allowed?
shb_hardware	2	variable	no
shb_os	3	variable	no
shb_userappl	4	variable	no

Table 2: Section Header Block Options

shb_hardware:

The shb_hardware option is a UTF-8 string containing the description of the hardware used to create this section. The string is not zero-terminated.

Examples: "x86 Personal Computer", "Sun Sparc Workstation".

shb_os:

The shb_os option is a UTF-8 string containing the name of the operating system used to create this section. The string is not zero-terminated.

Examples: "Windows XP SP2", "openSUSE 10.2".

shb_userappl:

The shb_userappl option is a UTF-8 string containing the name of the application used to create this section. The string is not zero-terminated.

Examples: "dumpcap V0.99.7".

[Open issue: does a program which re-writes a capture file change the original hardware/os/application info?]

4.2. Interface Description Block

An Interface Description Block (IDB) is the container for information describing an interface on which packet data is captured.

Tools that write / read the capture file associate an incrementing unsigned 32-bit number (starting from '0') to each Interface Definition Block, called the Interface ID for the interface in question. This number is unique within each Section and identifies

the interface to which the IDB refers; it is only unique inside the current section, so, two Sections can have different interfaces identified by the same Interface ID values. This unique identifier is referenced by other blocks, such as Enhanced Packet Blocks and Interface Statistic Blocks, to indicate the interface to which the block refers (such the interface that was used to capture the packet that an Enhanced Packet Block contains or to which the statistics in an Interface Statistic Block refer).

There must be an Interface Description Block for each interface to which another block refers. Blocks such as an Enhanced Packet Block or an Interface Statistics Block contain an Interface ID value referring to a particular interface, and a Simple Packet Block implicitly refers to an interface with an Interface ID of 0. If the file does not contain any blocks that use an Interface ID, then the file does not need to have any IDBs.

An Interface Description Block is valid only inside the section to which it belongs. The structure of a Interface Description Block is shown in Figure 10.

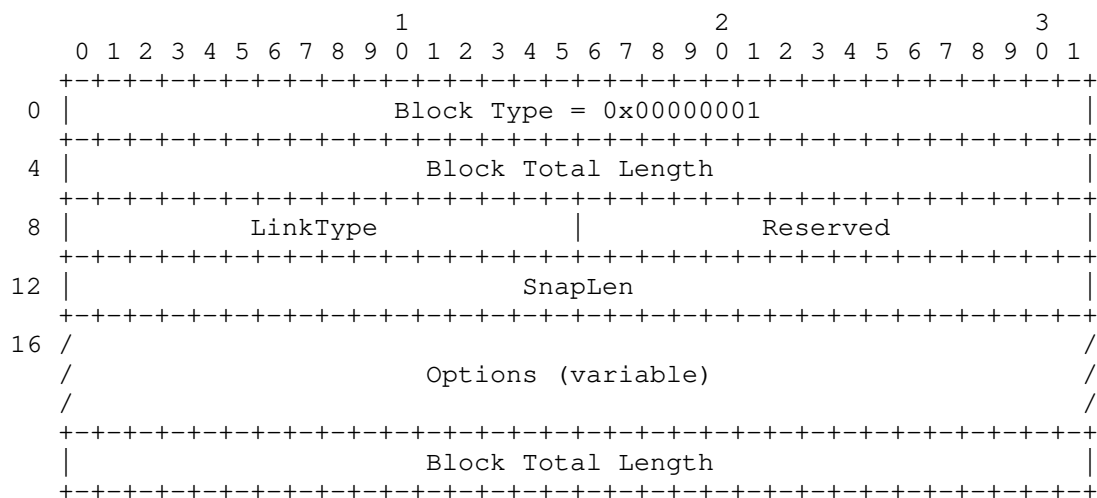


Figure 10: Interface Description Block Format

The meaning of the fields is:

- * Block Type: The block type of the Interface Description Block is 1.
- * Block Total Length: total size of this block, as described in Section 3.1.

- * **LinkType (16 bits):** an unsigned value that defines the link layer type of this interface. The list of Standardized Link Layer Type codes is available in [LINKTYPES].
- * **Reserved (16 bits):** not used - MUST be filled with 0 by pcapng file writers, and MUST be ignored by pcapng file readers.
- * **SnapLen (32 bits):** an unsigned value indicating the maximum number of octets captured from each packet. The portion of each packet that exceeds this value will not be stored in the file. A value of zero indicates no limit.
- * **Options:** optionally, a list of options (formatted according to the rules defined in Section 3.5) can be present.

In addition to the options defined in Section 3.5, the following options are valid within this block:

Name	Code	Length	Multiple allowed?
if_name	2	variable	no
if_description	3	variable	no
if_IPv4addr	4	8	yes
if_IPv6addr	5	17	yes
if_MACaddr	6	6	no
if_EUIaddr	7	8	no
if_speed	8	8	no
if_tsresol	9	1	no
if_tzone	10	4	no
if_filter	11	variable, minimum 1	no
if_os	12	variable	no
if_fcslen	13	1	no
if_tsoffset	14	8	no
if_hardware	15	variable	no
if_txspeed	16	8	no
if_rxspeed	17	8	no

Table 3: Interface Description Block Options

if_name:

The if_name option is a UTF-8 string containing the name of the device used to capture data. The string is not zero-terminated.

Examples: "eth0",

"\Device\NPF_{AD1CE675-96D0-47C5-ADD0-2504B9126B68}"/>.

if_description:

The `if_description` option is a UTF-8 string containing the description of the device used to capture data. The string is not zero-terminated.

Examples: "Wi-Fi", "Local Area Connection", "Wireless Network Connection", "First Ethernet Interface".

`if_IPv4addr`:

The `if_IPv4addr` option is an IPv4 network address and corresponding netmask for the interface. The first four octets are the IP address, and the next four octets are the netmask. This option can be repeated multiple times within the same Interface Description Block when multiple IPv4 addresses are assigned to the interface. Note that the IP address and netmask are both treated as four octets, one for each octet of the address or mask; they are not 32-bit numbers, and thus the endianness of the SHB does not affect this field's value.

Examples: '192 168 1 1 255 255 255 0'.

`if_IPv6addr`:

The `if_IPv6addr` option is an IPv6 network address and corresponding prefix length for the interface. The first 16 octets are the IP address and the next octet is the prefix length. This option can be repeated multiple times within the same Interface Description Block when multiple IPv6 addresses are assigned to the interface.

Example: 2001:0db8:85a3:08d3:1319:8a2e:0370:7344/64 is written (in hex) as '20 01 0d b8 85 a3 08 d3 13 19 8a 2e 03 70 73 44 40'.

`if_MACaddr`:

The `if_MACaddr` option is the Interface Hardware MAC address (48 bits), if available.

Example: '00 01 02 03 04 05'.

`if_EUIaddr`:

The `if_EUIaddr` option is the Interface Hardware EUI address (64 bits), if available.

Example: '02 34 56 FF FE 78 9A BC'.

`if_speed`:

The `if_speed` option is a 64-bit unsigned value indicating the interface speed, in bits per second.

Example: the 64-bit decimal number 100000000 for 100Mbps.

if_tsresol:

The if_tsresol option identifies the resolution of timestamps. If the Most Significant Bit is equal to zero, the remaining bits indicates the resolution of the timestamp as a negative power of 10 (e.g. 6 means microsecond resolution, timestamps are the number of microseconds since 1970-01-01 00:00:00 UTC). If the Most Significant Bit is equal to one, the remaining bits indicates the resolution as negative power of 2 (e.g. 10 means 1/1024 of second). If this option is not present, a resolution of 10^{-6} is assumed (i.e. timestamps have the same resolution of the standard 'libpcap' timestamps).

Example: '6'.

if_tzone:

The if_tzone option identifies the time zone for GMT support (TODO: specify better).

Example: TODO: give a good example.

if_filter:

The if_filter option identifies the filter (e.g. "capture only TCP traffic") used to capture traffic. The first octet of the Option Data keeps a code of the filter used (e.g. if this is a libpcap string, or BPF bytecode, and more). More details about this format will be presented in Appendix XXX (TODO). (TODO: better use different options for different fields? e.g. if_filter_pcap, if_filter_bpf, ...)

Example: '00'"tcp port 23 and host 192.0.2.5".

if_os:

The if_os option is a UTF-8 string containing the name of the operating system of the machine in which this interface is installed. This can be different from the same information that can be contained by the Section Header Block (Section 4.1) because the capture can have been done on a remote machine. The string is not zero-terminated.

Examples: "Windows XP SP2", "openSUSE 10.2".

if_fcslen:

The if_fcslen option is an 8-bit unsigned integer value that specifies the length of the Frame Check Sequence (in bits) for this interface. For link layers whose FCS length can

change during time, the Enhanced Packet Block `epb_flags` Option can be used in each Enhanced Packet Block (see Section 4.3.1).

Example: '4'.

`if_tsoffset:`

The `if_tsoffset` option is a 64-bit signed integer value that specifies an offset (in seconds) that must be added to the timestamp of each packet to obtain the absolute timestamp of a packet. If the option is missing, the timestamps stored in the packet MUST be considered absolute timestamps. The time zone of the offset can be specified with the option `if_tzone`. TODO: won't a `if_tsoffset_low` for fractional second offsets be useful for highly synchronized capture systems?

Example: '1234'.

`if_hardware:`

The `if_hardware` option is a UTF-8 string containing the description of the interface hardware. The string is not zero-terminated.

Examples: "Broadcom NetXtreme", "Intel(R) PRO/1000 MT Network Connection", "NETGEAR WNA1000Mv2 N150 Wireless USB Micro Adapter".

`if_txspeed:`

The `if_txrxspeeds` option is a 64-bit unsigned value indicating the interface transmit speed in bits per second.

Example: the 64-bit decimal number 1024000 for 1024Kbps.

`if_rxspeed:`

The `if_rxspeed` option is a 64-bit unsigned value indicating the interface receive speed, in bits per second.

Example: the 64-bit decimal number 8192000 for 8192Kbps.

If the interface transmit speed and receive speed are the same, the `if_speed` option MUST be used and the `if_txspeed` and `if_rxspeed` options MUST NOT be used. If the transmit speed is unknown, the `if_speed` and `if_txspeed` options MUST NOT be used; if the receive speed is unknown, the `if_speed` and `if_rxspeed` options MUST NOT be used.

4.3. Enhanced Packet Block

An Enhanced Packet Block (EPB) is the standard container for storing the packets coming from the network. The Enhanced Packet Block is optional because packets can be stored either by means of this block or the Simple Packet Block, which can be used to speed up capture file generation; or a file may have no packets in it. The format of an Enhanced Packet Block is shown in Figure 11.

The Enhanced Packet Block is an improvement over the original, now obsolete, Packet Block (Appendix A):

- * it stores the Interface Identifier as a 32-bit integer value. This is a requirement when a capture stores packets coming from a large number of interfaces;
- * unlike the Packet Block (Appendix A), the number of packets dropped by the capture system between this packet and the previous one is not stored in the header, but rather in an option of the block itself.

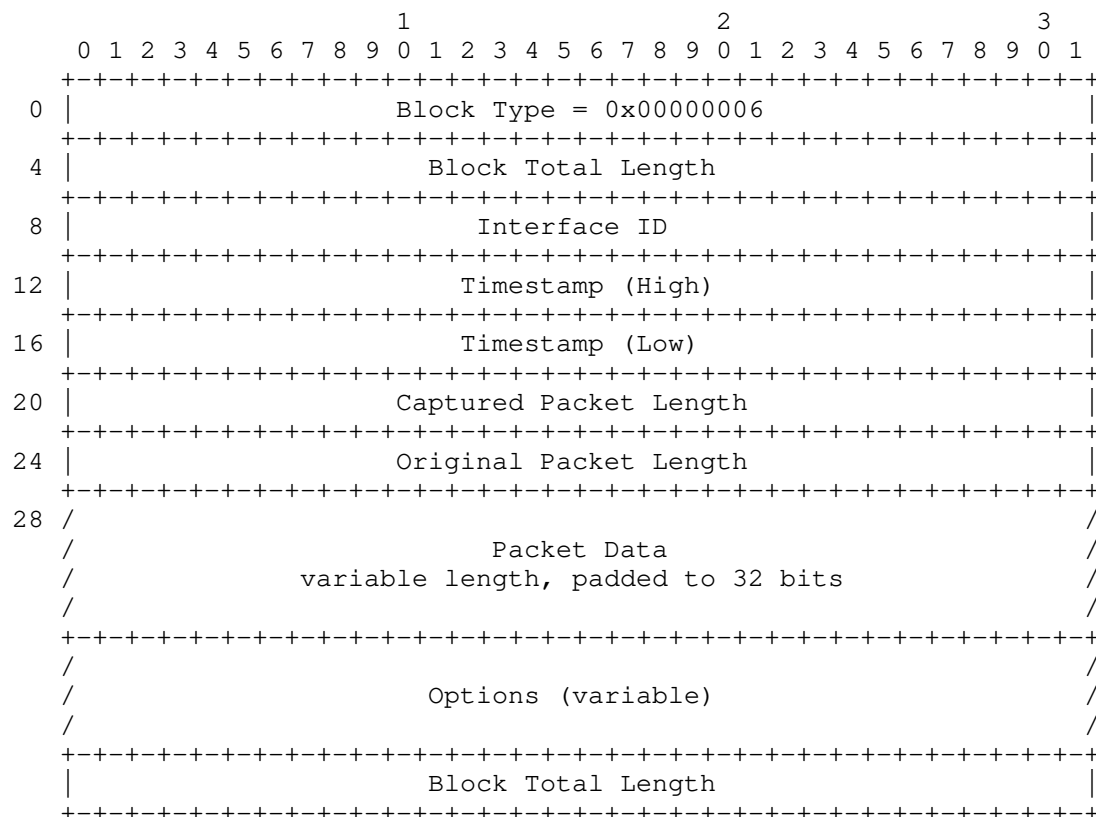


Figure 11: Enhanced Packet Block Format

The Enhanced Packet Block has the following fields:

- * Block Type: The block type of the Enhanced Packet Block is 6.
- * Block Total Length: total size of this block, as described in Section 3.1.
- * Interface ID (32 bits): an unsigned value that specifies the interface on which this packet was received or transmitted; the correct interface will be the one whose Interface Description Block (within the current Section of the file) is identified by the same number (see Section 4.2) of this field. The interface ID MUST be valid, which means that an matching interface description block MUST exist.

- * **Timestamp (High) and Timestamp (Low):** upper 32 bits and lower 32 bits of a 64-bit timestamp. The timestamp is a single 64-bit unsigned integer that represents the number of units of time that have elapsed since 1970-01-01 00:00:00 UTC. The length of a unit of time is specified by the 'if_tsresol' option (see Figure 10) of the Interface Description Block referenced by this packet. Note that, unlike timestamps in the libpcap file format, timestamps in Enhanced Packet Blocks are not saved as two 32-bit values that represent the seconds and microseconds that have elapsed since 1970-01-01 00:00:00 UTC. Timestamps in Enhanced Packet Blocks are saved as two 32-bit words that represent the upper and lower 32 bits of a single 64-bit quantity.
- * **Captured Packet Length (32 bits):** an unsigned value that indicates the number of octets captured from the packet (i.e. the length of the Packet Data field). It will be the minimum value among the Original Packet Length and the snapshot length for the interface (SnapLen, defined in Figure 10). The value of this field does not include the padding octets added at the end of the Packet Data field to align the Packet Data field to a 32-bit boundary.
- * **Original Packet Length (32 bits):** an unsigned value that indicates the actual length of the packet when it was transmitted on the network. It can be different from the Captured Packet Length if the packet has been truncated by the capture process.
- * **Packet Data:** the data coming from the network, including link-layer headers. The actual length of this field is Captured Packet Length plus the padding to a 32-bit boundary. The format of the link-layer headers depends on the LinkType field specified in the Interface Description Block (see Section 4.2) and it is specified in the entry for that format in [LINKTYPES].
- * **Options:** optionally, a list of options (formatted according to the rules defined in Section 3.5) can be present.

In addition to the options defined in Section 3.5, the following options are valid within this block:

Name	Code	Length	Multiple allowed?
epb_flags	2	4	no
epb_hash	3	variable, minimum hash type-dependent	yes
epb_dropcount	4	8	no
epb_packetid	5	8	no
epb_queue	6	4	no
epb_verdict	7	variable, minimum verdict type-dependent	yes

Table 4: Enhanced Packet Block Options

epb_flags:

The `epb_flags` option is a 32-bit flags word containing link-layer information. A complete specification of the allowed flags can be found in Section 4.3.1.

Example: '0'.

epb_hash:

The `epb_hash` option contains a hash of the packet. The first octet specifies the hashing algorithm, while the following octets contain the actual hash, whose size depends on the hashing algorithm, and hence from the value in the first octet. The hashing algorithm can be: 2s complement (algorithm octet = 0, size = XXX), XOR (algorithm octet = 1, size=XXX), CRC32 (algorithm octet = 2, size = 4), MD-5 (algorithm octet = 3, size = 16), SHA-1 (algorithm octet = 4, size = 20), Toeplitz (algorithm octet = 5, size = 4). The hash covers only the packet, not the header added by the capture driver: this gives the possibility to calculate it inside the network card. The hash allows easier comparison/merging of different capture files, and reliable data transfer between the data acquisition system and the capture library.

Examples: '02 EC 1D 87 97', '03 45 6E C2 17 7C 10 1E 3C 2E 99 6E C2 9A 3D 50 8E'.

epb_dropcount:

The `epb_dropcount` option is a 64-bit unsigned integer value specifying the number of packets lost (by the interface and the operating system) between this packet and the preceding one for the same interface or, for the first packet for an interface, between this packet and the start of the capture process.

Example: '0'.

epb_packetid:

The `epb_packetid` option is a 64-bit unsigned integer that uniquely identifies the packet. If the same packet is seen by multiple interfaces and there is a way for the capture application to correlate them, the same `epb_packetid` value must be used. An example could be a router that captures packets on all its interfaces in both directions. When a packet hits interface A on ingress, an EPB entry gets created, TTL gets decremented, and right before it egresses on interface B another EPB entry gets created in the trace file. In this case, two packets are in the capture file, which are not identical but the `epb_packetid` can be used to correlate them.

Example: '0'.

epb_queue:

The `epb_queue` option is a 32-bit unsigned integer that identifies on which queue of the interface the specific packet was received.

Example: '0'.

epb_verdict:

The `epb_verdict` option stores a verdict of the packet. The verdict indicates what would be done with the packet after processing it. For example, a firewall could drop the packet. This verdict can be set by various components, i.e. Hardware, Linux's eBPF TC or XDP framework, etc. etc. The first octet specifies the verdict type, while the following octets contain the actual verdict data, whose size depends on the verdict type, and hence from the value in the first octet. The verdict type can be: Hardware (type octet = 0, size = variable), Linux_eBPF_TC (type octet = 1, size = 8 (64-bit unsigned integer), value = TC_ACT_* as defined in the Linux `pck_cls.h` (https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/include/uapi/linux/pkt_cls.h include), Linux_eBPF_XDP (type octet = 2, size = 8 (64-bit

unsigned integer), value = xdp_action as defined in the Linux pbf.h (<https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/include/uapi/linux/bpf.h>) include).

Example: '02 00 00 00 00 00 00 00 02' for Linux_eBPF_XDP with verdict XDP_PASS.

4.3.1. Enhanced Packet Block Flags Word

The Enhanced Packet Block Flags Word is a 32-bit value that contains link-layer information about the packet.

The word is encoded as an unsigned 32-bit integer, using the endianness of the Section Header Block scope it is in. In the following table, the bits are numbered with 0 being the least-significant bit and 31 being the most-significant bit of the 32-bit unsigned integer. The meaning of the bits is the following:

Bit Number	Description
0-1	Inbound / Outbound packet (00 = information not available, 01 = inbound, 10 = outbound)
2-4	Reception type (000 = not specified, 001 = unicast, 010 = multicast, 011 = broadcast, 100 = promiscuous).
5-8	FCS length, in octets (0000 if this information is not available). This value overrides the if_fcslen option of the Interface Description Block, and is used with those link layers (e.g. PPP) where the length of the FCS can change during time.
9-15	Reserved (MUST be set to zero).
16-31	link-layer-dependent errors (Bit 31 = symbol error, Bit 30 = preamble error, Bit 29 = Start Frame Delimiter error, Bit 28 = unaligned frame error, Bit 27 = wrong Inter Frame Gap error, Bit 26 = packet too short error, Bit 25 = packet too long error, Bit 24 = CRC error, other?? are 16 bit enough?).

Table 5

NOTE: in earlier versions of this specification, the bits were specified as being numbered with 0 being the most-significant bit and 31 being the least-significant bit of the 32-bit unsigned integer, rather than with 0 being the least-significant bit and 31 being the most-significant bit. Several implementations number the bits with 0 being the least-significant bit, and no known implementations number them with 0 being the most-significant bit, so the specification was changed to reflect that reality.

4.4. Simple Packet Block

The Simple Packet Block (SPB) is a lightweight container for storing the packets coming from the network. Its presence is optional.

A Simple Packet Block is similar to an Enhanced Packet Block (see Section 4.3), but it is smaller, simpler to process and contains only a minimal set of information. This block is preferred to the standard Enhanced Packet Block when performance or space occupation are critical factors, such as in sustained traffic capture applications. A capture file can contain both Enhanced Packet Blocks and Simple Packet Blocks: for example, a capture tool could switch from Enhanced Packet Blocks to Simple Packet Blocks when the hardware resources become critical.

The Simple Packet Block does not contain the Interface ID field. Therefore, it MUST be assumed that all the Simple Packet Blocks have been captured on the interface previously specified in the first Interface Description Block.

Figure 12 shows the format of the Simple Packet Block.

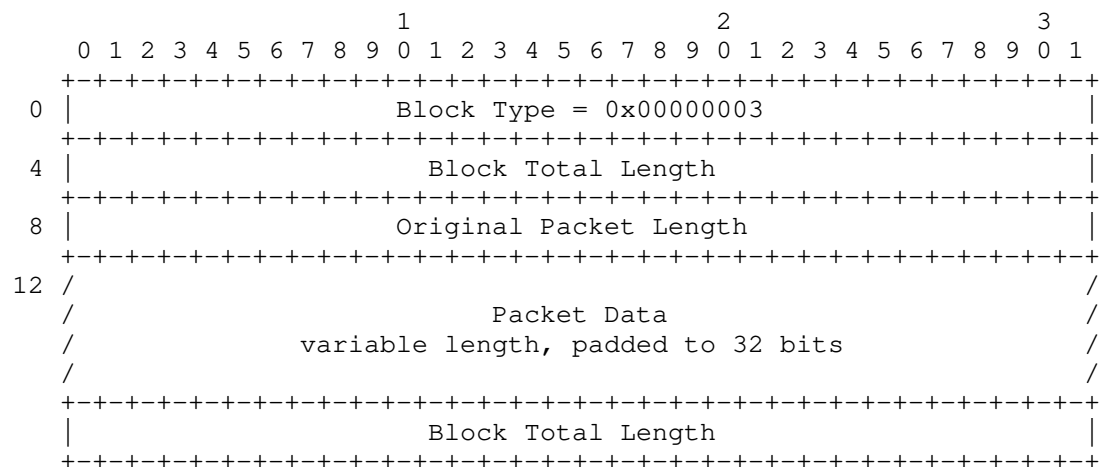


Figure 12: Simple Packet Block Format

The Simple Packet Block has the following fields:

- * Block Type: The block type of the Simple Packet Block is 3.
- * Block Total Length: total size of this block, as described in Section 3.1.
- * Original Packet Length (32 bits): an unsigned value indicating the actual length of the packet when it was transmitted on the network. It can be different from length of the Packet Data field's length if the packet has been truncated by the capture process, in which case the SnapLen value in Section 4.2 will be less than this Original Packet Length value, and the SnapLen value MUST be used to determine the size of the Packet Data field length.
- * Packet Data: the data coming from the network, including link-layer headers. The length of this field can be derived from the field Block Total Length, present in the Block Header, and it is the minimum value among the SnapLen (present in the Interface Description Block) and the Original Packet Length (present in this header). The format of the data within this Packet Data field depends on the LinkType field specified in the Interface Description Block (see Section 4.2) and it is specified in the entry for that format in [LINKTYPES].

The Simple Packet Block does not contain the timestamp because this is often one of the most costly operations on PCs. Additionally, there are applications that do not require it; e.g. an Intrusion Detection System is interested in packets, not in their timestamp.

A Simple Packet Block cannot be present in a Section that has more than one interface because of the impossibility to refer to the correct one (it does not contain any Interface ID field).

The Simple Packet Block is very efficient in term of disk space: a snapshot whose length is 100 octets requires only 16 octets of overhead, which corresponds to an efficiency of more than 86%.

4.5. Name Resolution Block

The Name Resolution Block (NRB) is used to support the correlation of numeric addresses (present in the captured packets) and their corresponding canonical names and it is optional. Having the literal names saved in the file prevents the need for performing name resolution at a later time, when the association between names and addresses may be different from the one in use at capture time. Moreover, the NRB avoids the need for issuing a lot of DNS requests every time the trace capture is opened, and also provides name resolution when reading the capture with a machine not connected to the network.

A Name Resolution Block is often placed at the beginning of the file, but no assumptions can be taken about its position. Multiple NRBs can exist in a pcapng file, either due to memory constraints or because additional name resolutions were performed by file processing tools, like network analyzers.

A Name Resolution Block need not contain any Records, except the `nrb_record_end` Record which MUST be the last Record. The addresses and names in NRB Records MAY be repeated multiple times; i.e., the same IP address may resolve to multiple names, the same name may resolve to the multiple IP addresses, and even the same address-to-name pair may appear multiple times, in the same NRB or across NRBs.

The format of the Name Resolution Block is shown in Figure 13.

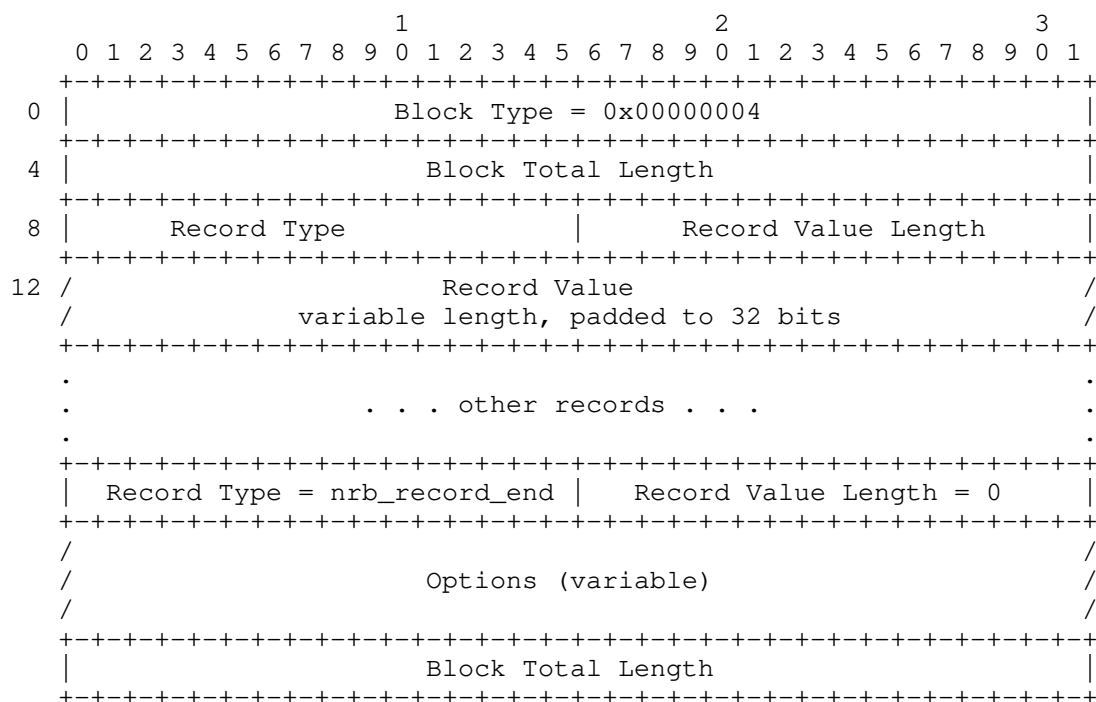


Figure 13: Name Resolution Block Format

The Name Resolution Block has the following fields:

- * Block Type: The block type of the Name Resolution Block is 4.
- * Block Total Length: total size of this block, as described in Section 3.1.

This is followed by zero or more Name Resolution Records (in the TLV format), each of which contains an association between a network address and a name. An `nrb_record_end` MUST be added after the last Record, and MUST exist even if there are no other Records in the NRB. There are currently three possible types of records:

Name	Code	Length
nrb_record_end	0x0000	0
nrb_record_ipv4	0x0001	variable
nrb_record_ipv6	0x0002	variable

Table 6: Name Resolution Block Records

nrb_record_end:

The nrb_record_end record delimits the end of name resolution records. This record is needed to determine when the list of name resolution records has ended and some options (if any) begin.

nrb_record_ipv4:

The nrb_record_ipv4 record specifies an IPv4 address (contained in the first 4 octets), followed by one or more zero-terminated UTF-8 strings containing the DNS entries for that address. The minimum valid Record Length for this Record Type is thus 6: 4 for the IP octets, 1 character, and a zero-value octet terminator. Note that the IP address is treated as four octets, one for each octet of the IP address; it is not a 32-bit word, and thus the endianness of the SHB does not affect this field's value.

Example: '127 0 0 1'"localhost".

[Open issue: is an empty string (i.e., just a zero-value octet) valid?]

nrb_record_ipv6:

The nrb_record_ipv6 record specifies an IPv6 address (contained in the first 16 octets), followed by one or more zero-terminated strings containing the DNS entries for that address. The minimum valid Record Length for this Record Type is thus 18: 16 for the IP octets, 1 character, and a zero-value octet terminator.

Example: '20 01 0d b8 00 00 00 00 00 00 00 00 12 34 56 78'"somehost".

[Open issue: is an empty string (i.e., just a zero-value octet) valid?]

Record Types other than those specified earlier MUST be ignored and skipped past. More Record Types will likely be defined in the future, and MUST NOT break backwards compatibility.

Each Record Value is aligned to and padded to a 32-bit boundary. The corresponding Record Value Length reflects the actual length of the Record Value; it does not include the lengths of the Record Type field, the Record Value Length field, any padding for the Record Value, or anything after the Record Value. For Record Types with name strings, the Record Length does include the zero-value octet terminating that string. A Record Length of 0 is valid, unless indicated otherwise.

After the list of Name Resolution Records, optionally, a list of options (formatted according to the rules defined in Section 3.5) can be present.

In addition to the options defined in Section 3.5, the following options are valid within this block:

Name	Code	Length	Multiple allowed?
ns_dnsname	2	variable	no
ns_dnsIP4addr	3	4	no
ns_dnsIP6addr	4	16	no

Table 7: Name Resolution Block Options

ns_dnsname:

The ns_dnsname option is a UTF-8 string containing the name of the machine (DNS server) used to perform the name resolution. The string is not zero-terminated.

Example: "our_nameserver".

ns_dnsIP4addr:

The ns_dnsIP4addr option specifies the IPv4 address of the DNS server. Note that the IP address is treated as four octets, one for each octet of the IP address; it is not a 32-bit word, and thus the endianness of the SHB does not affect this field's value.

Example: '192 168 0 1'.

ns_dnsIP6addr:

The ns_dnsIP6addr option specifies the IPv6 address of the DNS server.

Example: '20 01 0d b8 00 00 00 00 00 00 00 00 12 34 56 78'.

4.6. Interface Statistics Block

The Interface Statistics Block (ISB) contains the capture statistics for a given interface and it is optional. The statistics are referred to the interface defined in the current Section identified by the Interface ID field. An Interface Statistics Block is normally placed at the end of the file, but no assumptions can be taken about its position - it can even appear multiple times for the same interface.

The format of the Interface Statistics Block is shown in Figure 14.

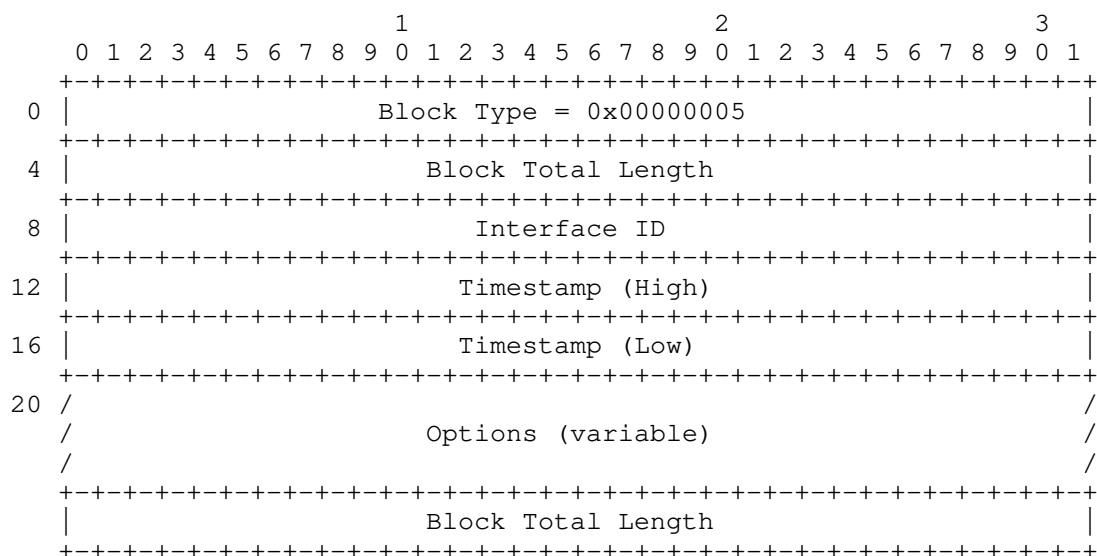


Figure 14: Interface Statistics Block Format

The fields have the following meaning:

- * Block Type: The block type of the Interface Statistics Block is 5.
- * Block Total Length: total size of this block, as described in Section 3.1.

- * Interface ID: specifies the interface these statistics refers to; the correct interface will be the one whose Interface Description Block (within the current Section of the file) is identified by same number (see Section 4.2) of this field.
- * Timestamp: time this statistics refers to. The format of the timestamp is the same already defined in the Enhanced Packet Block (Section 4.3); the length of a unit of time is specified by the 'if_tsresol' option (see Figure 10) of the Interface Description Block referenced by this packet.
- * Options: optionally, a list of options (formatted according to the rules defined in Section 3.5) can be present.

All the statistic fields are defined as options in order to deal with systems that do not have a complete set of statistics. Therefore, In addition to the options defined in Section 3.5, the following options are valid within this block:

Name	Code	Length	Multiple allowed?
isb_starttime	2	8	no
isb_endtime	3	8	no
isb_ifrecv	4	8	no
isb_ifdrop	5	8	no
isb_filteraccept	6	8	no
isb_osdrop	7	8	no
isb_usrdeliv	8	8	no

Table 8: Interface Statistics Block Options

isb_starttime:

The isb_starttime option specifies the time the capture started; time will be stored in two blocks of four octets each. The format of the timestamp is the same as the one defined in the Enhanced Packet Block (Section 4.3); the length of a unit of time is specified by the 'if_tsresol' option (see Figure 10) of the Interface Description Block referenced by this packet.

Example: '96 c3 04 00 73 89 6a 65', in Little Endian, decodes to 2012-06-29 06:17:00.834163 UTC.

isb_endtime:

The `isb_endtime` option specifies the time the capture ended; time will be stored in two blocks of four octets each. The format of the timestamp is the same as the one defined in the Enhanced Packet Block (Section 4.3); the length of a unit of time is specified by the 'if_tsresol' option (see Figure 10) of the Interface Description Block referenced by this packet.

Example: '97 c3 04 00 aa 47 ca 64', in Little Endian, decodes to 2012-06-29 07:28:25.298858 UTC.

isb_ifrecv:

The `isb_ifrecv` option specifies the 64-bit unsigned integer number of packets received from the physical interface starting from the beginning of the capture.

Example: the decimal number 100.

isb_ifdrop:

The `isb_ifdrop` option specifies the 64-bit unsigned integer number of packets dropped by the interface due to lack of resources starting from the beginning of the capture.

Example: '0'.

isb_filteraccept:

The `isb_filteraccept` option specifies the 64-bit unsigned integer number of packets accepted by filter starting from the beginning of the capture.

Example: the decimal number 100.

isb_osdrop:

The `isb_osdrop` option specifies the 64-bit unsigned integer number of packets dropped by the operating system starting from the beginning of the capture.

Example: '0'.

isb_usrdeliv:

The `isb_usrdeliv` option specifies the 64-bit unsigned integer number of packets delivered to the user starting from the beginning of the capture. The value contained in this field can be different from the value `'isb_filteraccept - isb_osdrop'` because some packets could still be in the OS buffers when the capture ended.

Example: `'0'`.

All the fields that refer to packet counters are 64-bit values, represented with the octet order of the current section. Special care must be taken in accessing these fields: since all the blocks are aligned to a 32-bit boundary, such fields are not guaranteed to be aligned on a 64-bit boundary.

4.7. Decryption Secrets Block

A Decryption Secrets Block (DSB) stores (session) secrets that enable decryption of packets within the capture file. The format of these secrets is defined by the Secrets Type.

Multiple DSBs can exist in a pcapng file, but they SHOULD be written before packet blocks that require those secrets. Tools MAY limit decryption to secrets that appear before packet blocks.

The structure of a Decryption Secrets Block is shown in Figure 15.

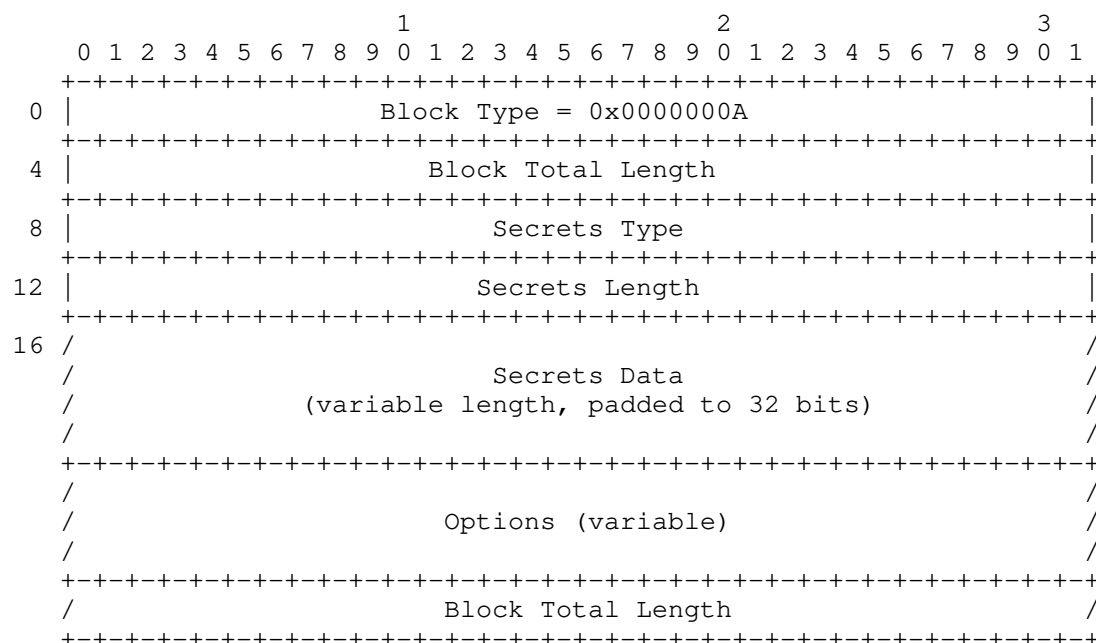


Figure 15: Decryption Secrets Block Format

The Decryption Secrets Block has the following fields.

- * Block Type: The block type of the Decryption Secrets Block is 10.
- * Block Total Length: total size of this block, as described in Section 3.1.
- * Secrets Type (32 bits): an unsigned integer identifier that describes the format of the following Secrets field. Requests for new Secrets Type codes should be made by creating a pull request to update this document as described in Section 10.1.
- * Secrets Length (32 bits): an unsigned integer that indicates the size of the following Secrets field, without any padding octets.
- * Secrets Data: binary data containing secrets, padded to a 32 bit boundary.
- * Options: optionally, a list of options (formatted according to the rules defined in Section 3.5) can be present. No DSB-specific options are currently defined.

The following is a list of Secrets Types.

0x544c534b:

TLS Key Log. This format is described at NSS Key Log Format (https://developer.mozilla.org/en-US/docs/Mozilla/Projects/NSS/Key_Log_Format). Every line MUST be properly terminated with either carriage return and linefeed ('`\r\n`') or linefeed ('`\n`'). Tools MUST be able to handle both line endings.

0x57474b4c:

WireGuard Key Log. Every line consists of the key type, equals sign ('`=`'), and the base64-encoded 32-byte key with optional spaces before and in between. The key type is one of `LOCAL_STATIC_PRIVATE_KEY`, `REMOTE_STATIC_PUBLIC_KEY`, `LOCAL_EPHEMERAL_PRIVATE_KEY`, or `PRESHARED_KEY`. This matches the output of `extract-handshakes.sh` (<https://git.zx2c4.com/WireGuard/tree/contrib/examples/extract-handshakes/README>), which is part of the WireGuard (<https://www.wireguard.com/>) project. A `PRESHARED_KEY` line is linked to a session matched by a previous `LOCAL_EPHEMERAL_PRIVATE_KEY` line. Every line MUST be properly terminated with either carriage return and linefeed ('`\r\n`') or linefeed ('`\n`'). Tools MUST be able to handle both line endings.

Warning: `LOCAL_STATIC_PRIVATE_KEY` and potentially `PRESHARED_KEY` are long-term secrets, users SHOULD only store non-production keys, or ensure proper protection of the pcapng file.

0x5a4e574b:

ZigBee NWK Key and ZigBee PANID for that network. Network Key as described in the ZigBee Specification (<https://zigbeealliance.org/>) 05-3473-21 (R21) section 4.2.2. The NWK Key is a 16 octet binary AES-128 key used to secure NWK Level frames within a single PAN. The NWK key is immediately followed by the 2 octet (16 bit) network PANID in little endian format. If and when the NWK Key changes a new DSB will contain the new NWK Key.

0x5a415053:

ZigBee APS Key. Application Support Link Key as described in the ZigBee Specification (<https://zigbeealliance.org/>) 05-3473-21 (R21) section 4.4. Each 16 octet binary AES-128 key secures frames exchanged between a pair of network nodes. The APS Key is immediately followed by the 2 octet (16 bit) network PANID in little endian format. The PANID is followed by the 2 octet (16 bit) short addresses, in little endian format, of the nodes to which the APS Key applies. The numerically lower short address shall come first. There is a

APS Key DSB for each node pair for which the Link Key is known. As new links are formed, new DSBs contain the new Keys. If the APS Key changes for an existing link, it is contained in a new DSB with the new APS Key.

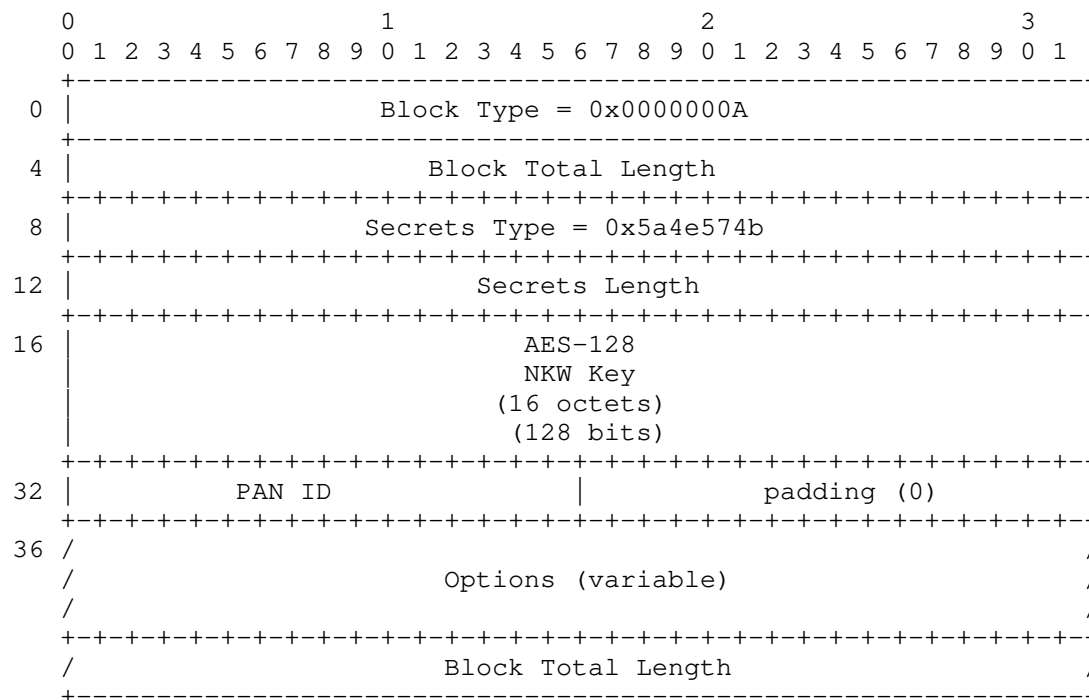


Figure 16: ZigBee NWK Key Data Format

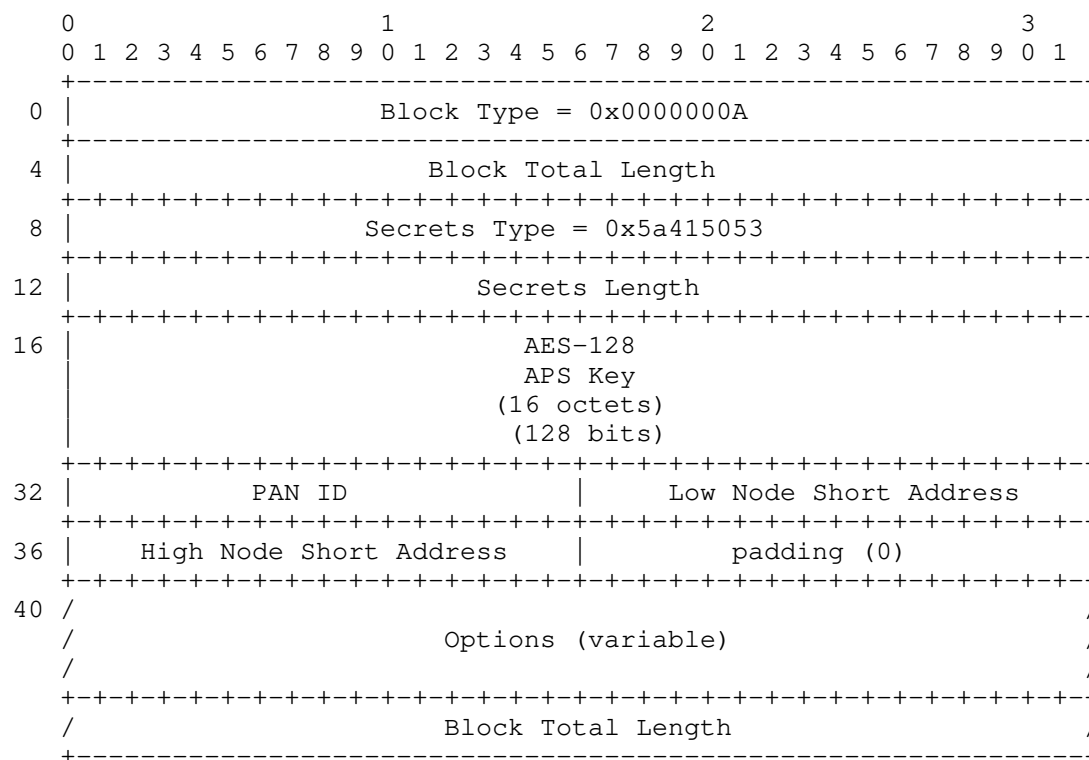


Figure 17: ZigBee APS Key Data Format

4.8. Custom Block

A Custom Block (CB) is the container for storing custom data that is not part of another block; for storing custom data as part of another block, see Section 3.5.1. The Custom Block is optional, can be repeated any number of times, and can appear before or after any other block except the first Section Header Block which must come first in the file. Different Custom Blocks, of different type codes and/or different Private Enterprise Numbers, may be used in the same pcapng file. The format of a Custom Block is shown in Figure 18.

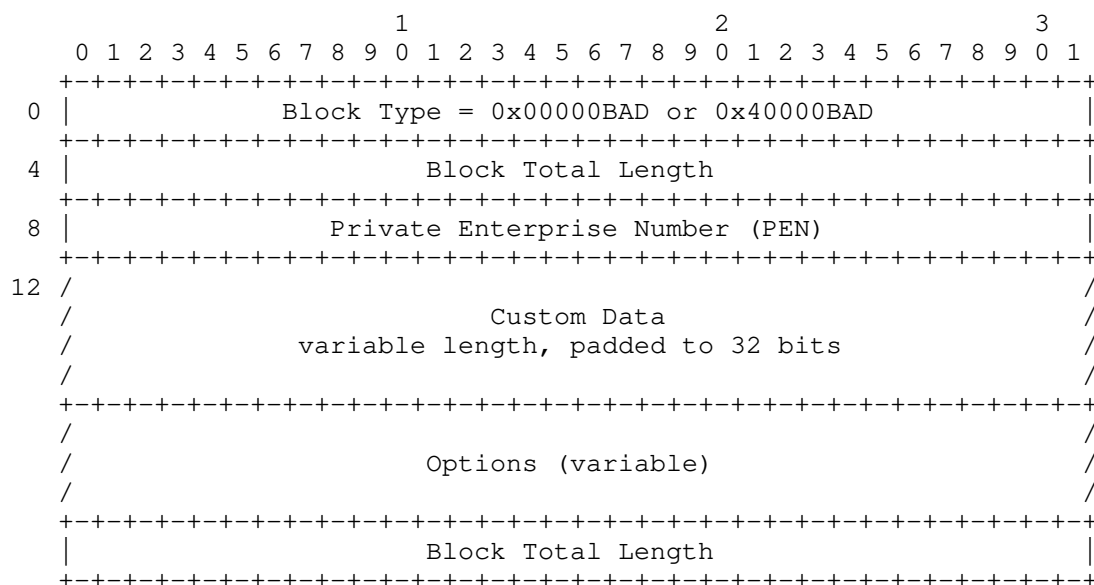


Figure 18: Custom Block Format

The Custom Block uses the type code 0x00000BAD (2989 in decimal) for a custom block that pcapng re-writers can copy into new files, and the type code 0x40000BAD (1073744813 in decimal) for one that should not be copied. See Section 5.2 for details.

The Custom Block has the following fields:

- * **Block Type:** The block type of the Custom Block is 0x00000BAD or 0x40000BAD, as described previously.
- * **Block Total Length:** total size of this block, as described in Section 3.1.
- * **Private Enterprise Number (32 bits):** An IANA-assigned Private Enterprise Number identifying the organization which defined the Custom Block. See Section 5.1 for details. The PEN MUST be encoded using the same endianness as the Section Header Block it is within the scope of.
- * **Custom Data:** the custom data, padded to a 32 bit boundary.
- * **Options:** optionally, a list of options (formatted according to the rules defined in Section 3.5) can be present. Note that custom options for the Custom Block still use the custom option format and type code, as described in Section 3.5.1.

5. Vendor-Specific Custom Extensions

This section uses the term "vendor" to describe an organization which extends the pcapng file with custom, proprietary blocks or options. It should be noted, however, that the "vendor" is just an abstract entity that agrees on a custom extension format: for example it may be a manufacturer, industry association, an individual user, or collective group of users.

5.1. Supported Use-Cases

There are two different supported use-cases for vendor-specific custom extensions: local and portable. Local use means the custom data is only expected to be usable on the same machine, and the same application, which encoded it into the file. This limitation is due to the lack of a common registry for the local use number codes (the block or option type code numbers with the Most Significant Bit set). Since two different vendors may choose the same number, one vendor's application reading the other vendor's file would result in decoding failure. Therefore, vendors SHOULD instead use the portable method, as described next.

The portable use-case supports vendor-specific custom extensions in pcapng files which can be shared across systems, organizations, etc. To avoid number space collisions, an IANA-registered Private Enterprise Number (PEN) is encoded into the Custom Block or Custom Option, using the PEN that belongs to the vendor defining the extension. Anyone can register a new PEN with IANA, for free, by filling out the online request form at [http://pen.iana.org/pen/](http://pen.iana.org/pen/PenApplication.page) PenApplication.page (<http://pen.iana.org/pen/PenApplication.page>).

5.2. Controlling Copy Behavior

Both Custom Blocks and Custom Options support two different codes to distinguish their "copy" behavior: a code for when the block or option can be safely copied into a new pcapng file by a pcapng manipulating application, and a code for when it should not be copied. A common reason for not copying a Custom Block or Custom Option is because it depends on other blocks or options in some way that would invalidate the custom data if the other blocks/options were removed or re-ordered. For example, if a Custom Block's data includes an Interface ID number in its Custom Data portion, then it cannot be safely copied by a pcapng application that merges pcapng files, because the merging application might re-order or remove one or more of the Interface Description Blocks, and thereby change the Interface IDs that the Custom Block depends upon. The same issue arises if a Custom Block or Custom Option depends on the presence of, or specific ordering of, other standard-based or custom-defined blocks or options.

Note that the copy semantics is not related to privacy - there is no guarantee that a pcapng anonymizer will remove a Custom Block or Custom Option, even if the appropriate code is used requesting it not be copied; and the original pcapng file can be shared anyway. If the Custom Data portion of the Custom Block or Custom Option contains sensitive information, then it should be encrypted in some fashion.

5.3. Strings vs. Octets

For the Custom Options, there are two Custom Data formats supported: a UTF-8 string and a binary data payload. The rationale for this separation is that a pcapng display application which does not understand the specific PEN's Custom Option can still display the data as a string if it's a string type code, rather than as hex-ascii of the octets.

5.4. Endianness Issues

Implementers writing Custom Blocks or binary data Custom Options should be aware that a pcapng file can be re-written by machines using a different endianness than the original file, which means all known fields of the pcapng file will change endianness in the new file. Since the Custom Data payload of the Custom Block or the binary data Custom Option might be an arbitrary sequence of unknown octets to such machines, they cannot convert multi-octet values inside the Custom Data, or in the Options section of a Custom Block, into the appropriate endianness.

For example, a little-endian machine can create a new pcapng file and add some binary data Custom Options to some non-Custom Block(s) in the file. This file can then be sent to a big-endian host, which will convert the Option Code, Option Length, and PEN fields of the options to big-endian format if it re-writes the file. However, if the software reading the file does not understand the contents of all of the Custom Options, it will leave the Custom Data payload of the options alone (as little-endian format). If this file then gets sent to a little-endian machine, then, when that little-endian machine reads the file, it will, if the software reading the file understands the contents of all the Custom Options, it will detect that the file format is big-endian, and swap the endianness while it parses the file - but that will cause the Custom Data payload to be incorrect since it was already in little-endian format.

In addition, a little-endian machine can create a pcapng file and write some binary data Custom Blocks, containing options, to the file. The file can then be sent to a big-endian host, which, if the software reading the file does not understand the contents of the Custom Blocks, will leave the Custom Data and Options alone (as little-endian format). If this file then gets sent to a little-endian machine, then, when that little-endian machine reads the file, it will, if the software reading the file understands the contents of all the Custom Blocks, it will detect that the file format is big-endian, and swap the endianness while it parses the file - but that will cause the Custom Data payload, the Option Code and Option Length values in the Options, and the PEN in any Custom Options to be incorrect since they were already in little-endian format.

Therefore, the vendor should either encode the Custom Data of their Custom Blocks and Custom Options, the Option Code and Option Length fields of options in Custom Blocks, and the PEN field of Custom Options in Custom Blocks in a consistent manner, such as always in big-endian or always in little-endian format, regardless of the host platform's endianness, or should encode some flag in the Custom Data payload to indicate in which endianness the rest of the payload is written.

The PEN field of a Custom Block, or of a Custom Option not contained in a Custom Block, MUST be converted by code that reads pcapng files, so this is not an issue for that field, except for Custom Options in Custom Blocks. This is also not an issue for the Custom Data payload of UTF-8 string Custom Options.

6. Recommended File Name Extension: .pcapng

The recommended file name extension for the "PCAP Next Generation Capture File Format" specified in this document is ".pcapng".

On Windows and macOS, files are distinguished by an extension to their filename. Such an extension is technically not actually required, as applications should be able to automatically detect the pcapng file format through the "magic bytes" at the beginning of the file, as some other UN*X desktop environments do. However, using name extensions makes it easier to work with files (e.g. visually distinguish file formats) so it is recommended - though not required - to use .pcapng as the name extension for files following this specification.

Please note: To avoid confusion (such as the current usage of .cap for a plethora of different capture file formats) file name extensions other than .pcapng should be avoided.

7. Conclusions

The file format proposed in this document should be very versatile and satisfy a wide range of applications. In the simplest case, it can contain a raw capture of the network data, made of a series of Simple Packet Blocks. In the most complex case, it can be used as a repository for heterogeneous information. In every case, the file remains easy to parse and an application can always skip the data it is not interested in; at the same time, different applications can share the file, and each of them can benefit of the information produced by the others. Two or more files can be concatenated obtaining another valid file.

8. Implementations

Some known implementations that read or write the pcapng file format are listed on the pcapng GitHub wiki (<https://github.com/pcapng/pcapng/wiki/Implementations>).

9. Security Considerations

TBD.

10. IANA Considerations

TBD.

[Open issue: decide whether the block types, option types, NRB Record types, etc. should be IANA registries. And if so, what the IANA policy for each should be (see RFC 5226)]

10.1. Standardized Block Type Codes

Every Block is uniquely identified by a 32-bit integer value, stored in the Block Header.

As pointed out in Section 3.1, Block Type codes whose Most Significant Bit (bit 31) is set to 1 are reserved for local use by the application.

All the remaining Block Type codes (0x00000000 to 0x7FFFFFFF) are standardized by this document. Requests for new Block Type codes, Option Type codes, and Secrets Type codes should be made by creating a pull request to update this document at github.com/pcapng/pcapng (<https://github.com/pcapng/pcapng>). The pull request should add a description of the new block, option, or secret type to Section 4. The pull request description should contain a clear request for a new type code assignment.

The following is a list of the Standardized Block Type Codes:

Block Type Code	Description
0x00000000	Reserved ???
0x00000001	Interface Description Block (Section 4.2)
0x00000002	Packet Block (Appendix A)
0x00000003	Simple Packet Block (Section 4.4)
0x00000004	Name Resolution Block (Section 4.5)
0x00000005	Interface Statistics Block (Section 4.6)
0x00000006	Enhanced Packet Block (Section 4.3)
0x00000007	IRIG Timestamp Block (requested by Gianluca Varenni < gianluca.varenni@cacetech.com >, CACE Technologies LLC); code also used for Socket Aggregation Event Block (https://github.com/google/linux-sensor/blob/master/hone-pcapng.txt)
0x00000008	ARINC 429 (https://en.wikipedia.org/wiki/ARINC_429) in AFDX Encapsulation Information Block (requested by Gianluca Varenni < gianluca.varenni@cacetech.com >, CACE

	Technologies LLC)
0x00000009	[systemd Journal Export Block] [I-D.richardson-opsawg-pcapng-extras]
0x0000000A	Decryption Secrets Block (Section 4.7)
0x00000101	Hone Project (https://github.com/HoneProject) Machine Info Block (https://github.com/HoneProject/Linux-Sensor/wiki/Augmented-PCAP-Next-Generation-Dump-File-Format) (see also Google version (https://github.com/google/linux-sensor/blob/master/hone-pcapng.txt))
0x00000102	Hone Project (https://github.com/HoneProject) Connection Event Block (https://github.com/HoneProject/Linux-Sensor/wiki/Augmented-PCAP-Next-Generation-Dump-File-Format) (see also Google version (https://github.com/google/linux-sensor/blob/master/hone-pcapng.txt))
0x00000201	Sysdig (https://github.com/draios/sysdig) Machine Info Block
0x00000202	Sysdig (https://github.com/draios/sysdig) Process Info Block, version 1
0x00000203	Sysdig (https://github.com/draios/sysdig) FD List Block
0x00000204	Sysdig (https://github.com/draios/sysdig) Event Block
0x00000205	Sysdig (https://github.com/draios/sysdig) Interface List Block
0x00000206	Sysdig (https://github.com/draios/sysdig) User List Block
0x00000207	Sysdig (https://github.com/draios/sysdig) Process Info Block, version 2
0x00000208	Sysdig (https://github.com/draios/sysdig) Event Block with flags
0x00000209	Sysdig (https://github.com/draios/sysdig)

	Process Info Block, version 3
0x00000210	Sysdig (https://github.com/draios/sysdig) Process Info Block, version 4
0x00000211	Sysdig (https://github.com/draios/sysdig) Process Info Block, version 5
0x00000212	Sysdig (https://github.com/draios/sysdig) Process Info Block, version 6
0x00000213	Sysdig (https://github.com/draios/sysdig) Process Info Block, version 7
0x00000BAD	Custom Block that rewriters can copy into new files (Section 4.8)
0x40000BAD	Custom Block that rewriters should not copy into new files (Section 4.8)
0x0A0D0D0A	Section Header Block (Section 4.1)
0x0A0D0A00-0x0A0D0AFF	Reserved. Used to detect trace files corrupted because of file transfers using the HTTP protocol in text mode.
0x000A0D0A-0xFF0A0D0A	Reserved. Used to detect trace files corrupted because of file transfers using the HTTP protocol in text mode.
0x000A0D0D-0xFF0A0D0D	Reserved. Used to detect trace files corrupted because of file transfers using the HTTP protocol in text mode.
0x0D0D0A00-0x0D0D0AFF	Reserved. Used to detect trace files corrupted because of file transfers using the FTP protocol in text mode.
0x80000000-0xFFFFFFFF	Reserved for local use.

Table 9: Standardized Block Type Codes

[Open issue: reserve 0x40000000-0x7FFFFFFF for do-not-copy-bit range of base types?]

11. Contributors

Loris Degioanni and Gianluca Varenni were coauthoring this document before it was submitted to the IETF.

12. Acknowledgments

The authors wish to thank Anders Broman, Ulf Lamping, Richard Sharpe and many others for their invaluable comments.

13. References

13.1. 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>>.

13.2. Informative References

- [I-D.richardson-opsawg-pcapng-extras]
"*** BROKEN REFERENCE ***".
- [LINKTYPES]
The Tcpdump Group, "the tcpdump.org link-layer header types registry", <<http://www.tcpdump.org/linktypes.html>>.

Appendix A. Packet Block (obsolete!)

The Packet Block is obsolete, and MUST NOT be used in new files. Use the Enhanced Packet Block or Simple Packet Block instead. This section is for historical reference only.

A Packet Block was a container for storing packets coming from the network.

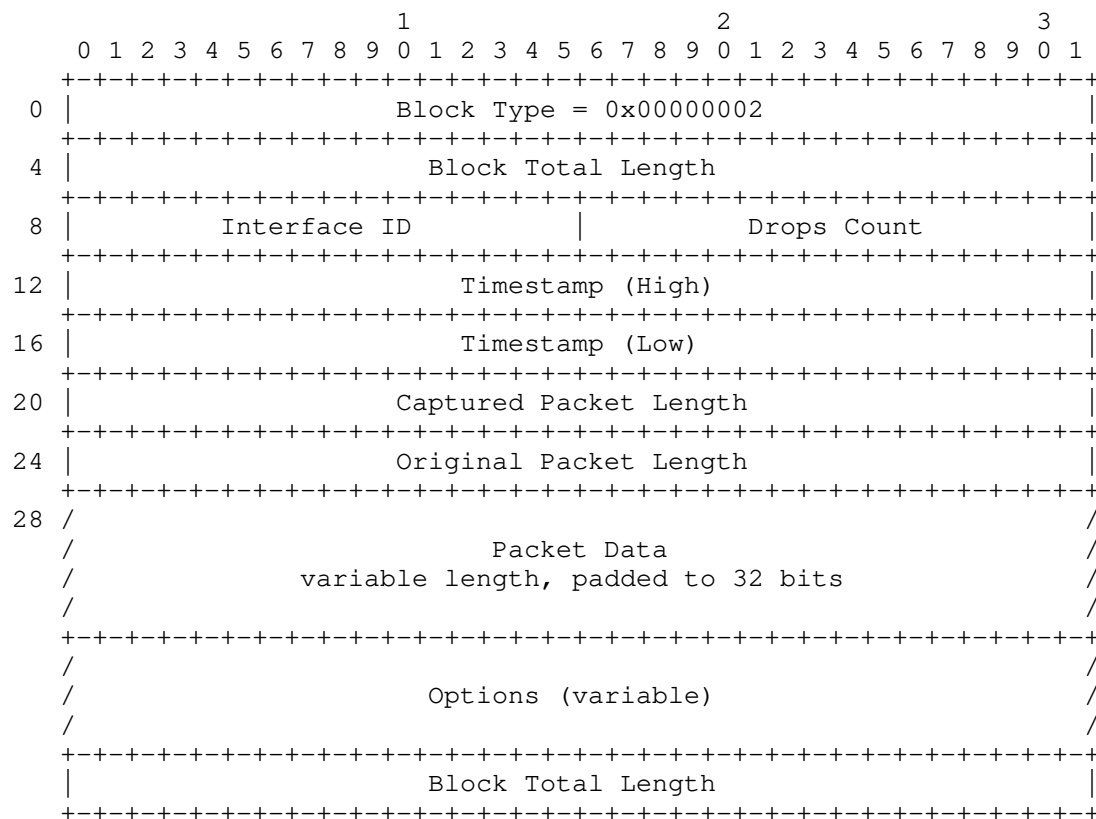


Figure 19: Packet Block Format

The Packet Block has the following fields:

- * Block Type: The block type of the Packet Block is 2.
- * Block Total Length: total size of this block, as described in Section 3.1.
- * Interface ID: specifies the interface this packet comes from; the correct interface will be the one whose Interface Description Block (within the current Section of the file) is identified by the same number (see Section 4.2) of this field. The interface ID MUST be valid, which means that an matching interface description block MUST exist.

- * **Drops Count:** a local drop counter. It specifies the number of packets lost (by the interface and the operating system) between this packet and the preceding one. The value xFFFF (in hexadecimal) is reserved for those systems in which this information is not available.
- * **Timestamp (High) and Timestamp (Low):** timestamp of the packet. The format of the timestamp is the same as was already defined for the Enhanced Packet Block (Section 4.3).
- * **Captured Packet Length:** number of octets captured from the packet (i.e. the length of the Packet Data field). It will be the minimum value among the Original Packet Length and the snapshot length for the interface (SnapLen, defined in Figure 10). The value of this field does not include the padding octets added at the end of the Packet Data field to align the Packet Data field to a 32-bit boundary.
- * **Original Packet Length:** actual length of the packet when it was transmitted on the network. It can be different from Captured Packet Length if the packet has been truncated by the capture process.
- * **Packet Data:** the data coming from the network, including link-layer headers. The actual length of this field is Captured Packet Length plus the padding to a 32-bit boundary. The format of the link-layer headers depends on the LinkType field specified in the Interface Description Block (see Section 4.2) and it is specified in the entry for that format in [LINKTYPES].
- * **Options:** optionally, a list of options (formatted according to the rules defined in Section 3.5) can be present.

In addition to the options defined in Section 3.5, the following options were valid within this block:

Name	Code	Length	Multiple allowed?
pack_flags	2	4	no
pack_hash	3	variable	yes

Table 10: Packet Block Options

pack_flags:

The `pack_flags` option is the same as the `epb_flags` of the enhanced packet block.

Example: `'0'`.

`pack_hash`:

The `pack_hash` option is the same as the `epb_hash` of the enhanced packet block.

Examples: `'02 EC 1D 87 97'`, `'03 45 6E C2 17 7C 10 1E 3C 2E 99 6E C2 9A 3D 50 8E'`.

Authors' Addresses

Michael Tuexen (editor)
Muenster University of Applied Sciences
Stegerwaldstrasse 39
48565 Steinfurt
Germany

Email: tuexen@fh-muenster.de

Fulvio Risso
Politecnico di Torino
Corso Duca degli Abruzzi, 24
10129 Torino
Italy

Email: fulvio.risso@polito.it

Jasper Bongertz
Airbus Defence and Space CyberSecurity
Kanzlei 63c
40667 Meerbusch
Germany

Email: jasper@packet-foo.com

Gerald Combs
Wireshark Foundation
339 Madson Pl
Davis, CA 95618
United States of America

Email: gerald@wireshark.org

Guy Harris

Email: gharris@sonic.net

Eelco Chaudron
Red Hat
De Entree 238
1101 EE Amsterdam
Netherlands

Email: eelco@redhat.com

Michael C. Richardson
Sandelman Software Works

Email: mcr+ietf@sandelman.ca
URI: <http://www.sandelman.ca/>

Internet Engineering Task Force
Internet-Draft
Updates: 6353 (if approved)
Intended status: Standards Track
Expires: 29 December 2021

K. Vaughn, Ed.
Trevilon LLC
27 June 2021

Transport Layer Security Version 1.3 (TLS 1.3) Transport Model for the
Simple Network Management Protocol Version 3 (SNMPv3)
draft-vaughn-tlstm-update-01

Abstract

This document updates the TLS Transport Model (TLSTM), as defined in [RFC6353], to support Transport Layer Security Version 1.3 (TLS) [RFC8446] and Datagram Transport Layer Security Version 1.3 (DTLS) [I-D.ietf-tls-dtls13], which are jointly known as "(D)TLS". This document may be applicable to future versions of SNMP and (D)TLS.

This document updates the SNMP-TLS-TM-MIB as defined in [RFC6353].

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 29 December 2021.

Copyright Notice

Copyright (c) 2021 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 Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Conventions	3
2. Changes from RFC 6353	4
2.1. TLSTM Fingerprint	4
2.2. Security Level	5
2.3. TLS Version	5
2.4. SNMP Version	5
2.5. Common Name	6
3. Additional Rules for TLS 1.3	6
3.1. Zero Round Trip Time Resumption (0-RTT)	6
3.2. TLS ciphersuites, extensions and protocol invariants . .	6
4. MIB Module Definition	6
5. Security Considerations	37
5.1. MIB Module Security	37
6. IANA Considerations	38
7. Acknowledgements	39
8. References	39
8.1. Normative References	39
8.2. Informative References	40
Appendix A. Target and Notification Configuration Example . . .	41
A.1. Configuring a Notification Originator	41
A.2. Configuring TLSTM to Utilize a Simple Derivation of tmSecurityName	42
A.3. Configuring TLSTM to Utilize Table-Driven Certificate Mapping	42
Author's Address	43

1. Introduction

This document updates the fingerprint algorithm defined by [RFC6353] to support the ciphersuites used by Transport Layer Security Version 1.3 (TLS) and Datagram Transport Layer Security Version 1.3 (DTLS), which are jointly known as "(D)TLS". The update also incorporates other less critical updates. Although the title and text of this document specifically reference SNMPv3 and (D)TLS 1.3, this document may be applicable to future versions of these protocols.

1.1. Conventions

Within this document the terms "TLS", "DTLS", "(D)TLS", "SNMP", and "TLSTM" mean "TLS 1.3", "DTLS 1.3", "TLS 1.3 and/or DTLS 1.3", "SNMPv3", and "TLSTM 1.3", respectively. These version numbers are only used when the text needs to emphasize version numbers, such as within the title. When this document refers to any other version of these protocols, it always explicitly states the version intended.

For consistency with SNMP-related specifications, this document favors terminology as defined in [STD62], rather than favoring terminology that is consistent with non-SNMP specifications. This is consistent with the IESG decision to not require the SNMPv3 terminology be modified to match the usage of other non-SNMP specifications when SNMPv3 was advanced to a Full Standard.

"Authentication" in this document typically refers to the English meaning of "serving to prove the authenticity of" the message, not data source authentication or peer identity authentication. The terms "manager" and "agent" are not used in this document because, in the RFC3411 architecture, all SNMP entities have the capability of acting as manager, agent, or both depending on the SNMP application types supported in the implementation. Where distinction is necessary, the application names of command generator, command responder, notification originator, notification receiver, and proxy forwarder are used. See "SNMP Applications" (RFC3411) for further information.

Throughout this document, the terms "client" and "server" are used to refer to the two ends of the TLS transport connection. The client actively opens the TLS connection, and the server passively listens for the incoming TLS connection. An SNMP entity MAY act as a TLS client or server or both, depending on the SNMP applications supported.

While TLS frequently refers to a user, the terminology preferred in RFC3411 and in this memo is "principal". A principal is the "who" on whose behalf services are provided or processing takes place. A principal can be, among other things, an individual acting in a particular role; a set of individuals, with each acting in a particular role; an application or a set of applications, or a combination of these within an administrative domain.

Throughout this document, the term "session" is used to refer to a secure association between two TLS Transport Models that permits the transmission of one or more SNMP messages within the lifetime of the session. The TLS protocol also has an internal notion of a session and although these two concepts of a session are related, when the term "session" is used this document is referring to the TLSTM's specific session and not directly to the TLS protocol's session.

The User-Based Security Model (USM) (RFC3414) is a mandatory-to-implement Security Model in [STD62]. The USM derives the securityName and securityLevel from the SNMP message received, even when the message was received over a secure transport. It is RECOMMENDED that deployments that support the TLSTM disable the USM, if it has been implemented.

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 [RFC2119].

2. Changes from RFC 6353

This document updates [RFC6353]. The changes from [RFC6353] are defined in the following clauses.

2.1. TLSTM Fingerprint

[RFC6353] defines a fingerprint algorithm that references the one-octet TLS 1.2 hash algorithm identifier. TLS 1.3 replaced the one-octet hash algorithm identifier with a two-octet TLS 1.3 cipher suite identifier thereby breaking the algorithm defined in [RFC6353]. The update to the SNMP-TLS-TM-MIB, as defined in Section 4, deprecates the original fingerprint TEXTUAL-CONVENTION and replaces it with a new TEXTUAL-CONVENTION.

The change also required an update to several objects within the tables defined within the SNMP-TLS-TM-MIB; further these objects are referenced by other (e.g., RowStatus) objects in a manner that requires deprecating and replacing the tables in their entirety. Thus, while the number of objects deprecated and replaced is significant the semantics of the changes are minor.

References to the older objects within [RFC6353] are applicable to the replacement objects. The newer objects are identified with names similar to those used in the original MIB but with a "13" inserted to reference TLS 1.3.

2.2. Security Level

The RFC3411 architecture recognizes three levels of security:

- * without authentication and without privacy (noAuthNoPriv)
- * with authentication but without privacy (authNoPriv)
- * with authentication and with privacy (authPriv)

With (D)TLS 1.3, authentication and privacy are always provided. Hence, all exchanges conforming to the rules of this document will include authentication and privacy, regardless of the security level requested.

```
// This is consistent with what was prescribed in RFC6353, where a
// TLS Transport Model is expected to provide for outgoing
// connections with a security level at least that of the requested
// security level.
```

2.3. TLS Version

[RFC6353] stated that TLSTM clients and servers MUST NOT request, offer, or use SSL 2.0. This document extends this statement such that TLSTM clients and servers MUST NOT request, offer, or use SSL 3.0, (D)TLSv 1.0, (D)TLS v1.1. See Appendix D.5 of [RFC8446] for further details. For backward compatibility issues with older TLS versions, see Appendix D of [RFC8446].

An implementation that supports these older protocols is not considered conformant to the TLSTM while the older protocols are enabled.

2.4. SNMP Version

[RFC6353] stated that using a non-transport-aware Security Model with a secure Transport Model was not recommended. This document tightens this statement such that TLSTM clients and servers MUST NOT request, offer, or use SNMPv1 or SNMPv2c message processing described in [RFC3584], or the User-based Security Model of SNMPv3.

An implementation that supports these older protocols is not considered conformant to the TLSTM while the older protocols are enabled.

2.5. Common Name

[RFC6353] stated that the use of a certificate's CommonName is deprecated and users were encouraged to use the subjectAltName. This document tightens this statement such that TLSTM clients and servers MUST NOT use the CommonName.

3. Additional Rules for TLS 1.3

This document specifies additional rules and clarifications for the use of TLS 1.3.

3.1. Zero Round Trip Time Resumption (0-RTT)

TLS 1.3 implementations for SNMPv3 MUST NOT enable the 0-RTT mode of session resumption (either sending or accepting) and MUST NOT automatically resend 0-RTT data if it is rejected by the server. The reason 0-RTT is disallowed is that there are no "safe" messages that if replayed will be guaranteed to cause no harm at a server side: all incoming notification or command responses are meant to be acted upon only once. See Security considerations section for further details.

TLS TM clients and servers MUST NOT request, offer or use the 0-RTT mode of TLS 1.3. [RFC8446] removed the renegotiation supported in TLS 1.2 [RFC5246]; for session resumption, it introduced a zero-RTT (0-RTT) mode, saving a round-trip at connection setup at the cost of increased risk of replay attacks (it is possible for servers to guard against this attack by keeping track of all the messages received). [RFC8446] requires a profile be written for any application that wants to use 0-RTT, specifying which messages are "safe to use" on this mode. The reason 0-RTT is disallowed here is that there are no "safe" SNMPv3 messages that if replayed will be sure to cause no harm at a server side: all incoming notification or command responses have consequences and are to be acted upon only once.

Renegotiation of sessions is not supported as it is not supported by TLS 1.3.

3.2. TLS ciphersuites, extensions and protocol invariants

[RFC8446] section 9 requires that, in the absence of application profiles, certain cipher suites, TLS extensions, and TLS protocol invariants are mandatory to implement. This document does not specify an application profile, hence all of the compliance requirements in [RFC8446] apply.

4. MIB Module Definition

```

SNMP-TLS-TM-MIB DEFINITIONS ::= BEGIN
IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE,
    OBJECT-IDENTITY, mib-2, snmpDomains,
    Counter32, Unsigned32, Gauge32, NOTIFICATION-TYPE
        FROM SNMPv2-SMI
        -- RFC 2578 or any update thereof
    TEXTUAL-CONVENTION, TimeStamp, RowStatus, StorageType,
    AutonomousType
        FROM SNMPv2-TC
        -- RFC 2579 or any update thereof
    MODULE-COMPLIANCE, OBJECT-GROUP, NOTIFICATION-GROUP
        FROM SNMPv2-CONF
        -- RFC 2580 or any update thereof
    SnmpAdminString
        FROM SNMP-FRAMEWORK-MIB
        -- RFC 3411 or any update thereof
    snmpTargetParamsName, snmpTargetAddrName
        FROM SNMP-TARGET-MIB
        -- RFC 3413 or any update thereof
;
snmpTlstmMIB MODULE-IDENTITY
    LAST-UPDATED "202106220000Z"

    ORGANIZATION "ISMS Working Group"
    CONTACT-INFO "Kenneth Vaughn
        Trevilon LLC
        6606 FM 1488 RD, STE 503
        Magnolia, TX 77354
        USA
        kvaughn@trevilon.com"

    DESCRIPTION "
        The TLS Transport Model MIB
        Copyright (c) 2010-2021 IETF Trust and the persons identified
        as authors of the code. All rights reserved.
        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject
        to the license terms contained in, the Simplified BSD License
        set forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (http://trustee.ietf.org/license-info)."
    REVISION      "202106220000Z"
    DESCRIPTION   "This version of this MIB module is part of
        RFC XXXX; see the RFC itself for full legal
        notices. This version updated the MIB to
        support (D)TLS 1.3."

    REVISION      "201107190000Z"
    DESCRIPTION   "This version of this MIB module is part of
        RFC 6353; see the RFC itself for full legal
        notices. The only change was to introduce
        new wording to reflect require changes for
        IDNA addresses in the SnmpTLSAddress TC."

```

```

    REVISION      "201005070000Z"
    DESCRIPTION   "This version of this MIB module is part of
                  RFC 5953; see the RFC itself for full legal
                  notices."
    ::= { mib-2 198 }
-- *****
-- subtrees of the SNMP-TLS-TM-MIB
-- *****
snmpTlstmNotifications OBJECT IDENTIFIER ::= { snmpTlstmMIB 0 }
snmpTlstmIdentities     OBJECT IDENTIFIER ::= { snmpTlstmMIB 1 }
snmpTlstmObjects        OBJECT IDENTIFIER ::= { snmpTlstmMIB 2 }
snmpTlstmConformance   OBJECT IDENTIFIER ::= { snmpTlstmMIB 3 }
-- *****
-- snmpTlstmObjects - Objects
-- *****
snmpTLSTCPDomain OBJECT-IDENTITY
    STATUS      current
    DESCRIPTION
        "The SNMP over TLS via TCP transport domain. The
        corresponding transport address is of type SnmpTLSAddress.
        The securityName prefix to be associated with the
        snmpTLSTCPDomain is 'tls'. This prefix may be used by
        security models or other components to identify which secure
        transport infrastructure authenticated a securityName."
    REFERENCE
        "RFC 2579: Textual Conventions for SMIV2"
    ::= { snmpDomains 8 }
snmpDTLSUDPDDomain OBJECT-IDENTITY
    STATUS      deprecated
    DESCRIPTION
        "The SNMP over DTLS via UDP transport domain. The
        corresponding transport address is of type SnmpTLSAddress.
        The securityName prefix to be associated with the
        snmpDTLSUDPDDomain is 'dtls'. This prefix may be used by
        security models or other components to identify which secure
        transport infrastructure authenticated a securityName."
    REFERENCE
        "RFC 2579: Textual Conventions for SMIV2"
    ::= { snmpDomains 9 }
SnmpTLSAddress ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "1a"
    STATUS      current
    DESCRIPTION
        "Represents an IPv4 address, an IPv6 address, or a
        US-ASCII-encoded hostname and port number.
        An IPv4 address must be in dotted decimal format followed by a
        colon ':' (US-ASCII character 0x3A) and a decimal port number
        in US-ASCII."

```

An IPv6 address must be a colon-separated format (as described in RFC 5952), surrounded by square brackets ('[', US-ASCII character 0x5B, and ']', US-ASCII character 0x5D), followed by a colon ':' (US-ASCII character 0x3A) and a decimal port number in US-ASCII.

A hostname is always in US-ASCII (as per RFC 1123); internationalized hostnames are encoded as A-labels as specified in RFC 5890. The hostname is followed by a colon ':' (US-ASCII character 0x3A) and a decimal port number in US-ASCII. The name SHOULD be fully qualified whenever possible.

Values of this textual convention may not be directly usable as transport-layer addressing information, and may require run-time resolution. As such, applications that write them must be prepared for handling errors if such values are not supported, or cannot be resolved (if resolution occurs at the time of the management operation).

The DESCRIPTION clause of TransportAddress objects that may have SnmpTLSAddress values must fully describe how (and when) such names are to be resolved to IP addresses and vice versa.

This textual convention SHOULD NOT be used directly in object definitions since it restricts addresses to a specific format. However, if it is used, it MAY be used either on its own or in conjunction with TransportAddressType or TransportDomain as a pair.

When this textual convention is used as a syntax of an index object, there may be issues with the limit of 128 sub-identifiers specified in SMIV2 (STD 58). It is RECOMMENDED that all MIB documents using this textual convention make explicit any limitations on index component lengths that management software must observe. This may be done either by including SIZE constraints on the index components or by specifying applicable constraints in the conceptual row DESCRIPTION clause or in the surrounding documentation."

REFERENCE

- "RFC 1123: Requirements for Internet Hosts - Application and Support
- RFC 5890: Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework
- RFC 5952: A Recommendation for IPv6 Address Text Representation

SYNTAX OCTET STRING (SIZE (1..255))
SnmpTLSFingerprint ::= TEXTUAL-CONVENTION
DISPLAY-HINT "1x:1x"
STATUS deprecated
DESCRIPTION

"A fingerprint value that can be used to uniquely reference other data of potentially arbitrary length.
An SnmpTLSTFingerprint value is composed of a 1-octet hashing algorithm identifier followed by the fingerprint value. The octet value encoded is taken from the IANA TLS HashAlgorithm Registry (RFC 5246). The remaining octets are filled using the results of the hashing algorithm.
This TEXTUAL-CONVENTION allows for a zero-length (blank) SnmpTLSTFingerprint value for use in tables where the fingerprint value may be optional. MIB definitions or implementations may refuse to accept a zero-length value as appropriate.
This textual convention was deprecated because TLS 1.3 uses a 2-octet cipher suite identifier rather than a 1-octet hashing algorithm identifier."

REFERENCE "RFC 5246: The Transport Layer
Security (TLS) Protocol Version 1.2
<http://www.iana.org/assignments/tls-parameters/>

"

SYNTAX OCTET STRING (SIZE (0..255))
SnmpTLS13Fingerprint ::= TEXTUAL-CONVENTION
DISPLAY-HINT "1x,1x"
STATUS current
DESCRIPTION
"A fingerprint value that can be used to uniquely reference other data of potentially arbitrary length.
An SnmpTLS13Fingerprint value is composed of a 2-octet cipher suite identifier followed by the fingerprint value. The octet value encoded is taken from the IANA TLS Cipher Suites Registry(RFC 8446). The remaining octets are filled using the results of the hashing algorithm, up to the first 253 octets.
This TEXTUAL-CONVENTION allows for a zero-length (blank) SnmpTLS13Fingerprint value for use in tables where the fingerprint value may be optional. MIB definitions or implementations may refuse to accept a zero-length value as appropriate."

REFERENCE "RFC 8446: The Transport Layer
Security (TLS) Protocol Version 1.3
<http://www.iana.org/assignments/tls-parameters/>

"

SYNTAX OCTET STRING (SIZE (0..255))
-- Identities for use in the snmpTlstmCertToTSNTable and
-- snmpTlstmCertToTSN13Table
snmpTlstmCertToTSNMIdentities OBJECT IDENTIFIER
::= { snmpTlstmIdentities 1 }
snmpTlstmCertSpecified OBJECT-IDENTITY
STATUS current
DESCRIPTION "Directly specifies the tmSecurityName to be used for

this certificate. The value of the tmSecurityName to use is specified in the snmpTlstmCertToTSN13Data column. The snmpTlstmCertToTSN13Data column must contain a non-zero length SnmpAdminString compliant value or the mapping described in this row must be considered a failure."

```

::= { snmpTlstmCertToTSNMIdentities 1 }
snmpTlstmCertSANRFC822Name OBJECT-IDENTITY
STATUS          current
DESCRIPTION     "Maps a subjectAltName's rfc822Name to a
tmSecurityName. The local part of the rfc822Name is
passed unaltered but the host-part of the name must
be passed in lowercase. This mapping results in a
1:1 correspondence between equivalent subjectAltName
rfc822Name values and tmSecurityName values except
that the host-part of the name MUST be passed in
lowercase.
Example rfc822Name Field:  FooBar@Example.COM
is mapped to tmSecurityName: FooBar@example.com."

::= { snmpTlstmCertToTSNMIdentities 2 }
snmpTlstmCertSANDNSName OBJECT-IDENTITY
STATUS          current
DESCRIPTION     "Maps a subjectAltName's dNSName to a
tmSecurityName after first converting it to all
lowercase (RFC 5280 does not specify converting to
lowercase so this involves an extra step). This
mapping results in a 1:1 correspondence between
subjectAltName dNSName values and the tmSecurityName
values."

REFERENCE "RFC 5280 - Internet X.509 Public Key Infrastructure
Certificate and Certificate Revocation
List (CRL) Profile."

::= { snmpTlstmCertToTSNMIdentities 3 }
snmpTlstmCertSANIpAddress OBJECT-IDENTITY
STATUS          current
DESCRIPTION     "Maps a subjectAltName's ipAddress to a
tmSecurityName by transforming the binary encoded
address as follows:
1) for IPv4, the value is converted into a
decimal-dotted quad address (e.g., '192.0.2.1').
2) for IPv6 addresses, the value is converted into a
32-character all lowercase hexadecimal string
without any colon separators.
This mapping results in a 1:1 correspondence between
subjectAltName ipAddress values and the
tmSecurityName values.
The resulting length of an encoded IPv6 address is
the maximum length supported by the View-Based

```

Access Control Model (VACM). Using both the Transport Security Model's support for transport prefixes (see the SNMP-TSM-MIB's `snmpTsmConfigurationUsePrefix` object for details) will result in `securityName` lengths that exceed what VACM can handle."

```
 ::= { snmpTlstmCertToTSNMIdentities 4 }
snmpTlstmCertSANAny OBJECT-IDENTITY
STATUS          current
DESCRIPTION     "Maps any of the following fields using the
                  corresponding mapping algorithms:
```

Type	Algorithm
<code>rfc822Name</code>	<code>snmpTlstmCertSANRFC822Name</code>
<code>dNSName</code>	<code>snmpTlstmCertSANDNSName</code>
<code>iPAddress</code>	<code>snmpTlstmCertSANIpAddress</code>

The first matching `subjectAltName` value found in the certificate of the above types MUST be used when deriving the `tmSecurityName`. The mapping algorithm specified in the 'Algorithm' column MUST be used to derive the `tmSecurityName`.

This mapping results in a 1:1 correspondence between `subjectAltName` values and `tmSecurityName` values. The three sub-mapping algorithms produced by this combined algorithm cannot produce conflicting results between themselves."

```
 ::= { snmpTlstmCertToTSNMIdentities 5 }
snmpTlstmCertCommonName OBJECT-IDENTITY
STATUS          deprecated
DESCRIPTION     "Maps a certificate's CommonName to a tmSecurityName
                  after converting it to a UTF-8 encoding. The usage
                  of CommonNames is deprecated and users are
                  encouraged to use subjectAltName mapping methods
                  instead. This mapping results in a 1:1
                  correspondence between certificate CommonName values
                  and tmSecurityName values."
 ::= { snmpTlstmCertToTSNMIdentities 6 }
```

```
-- The snmpTlstmSession Group
snmpTlstmSession          OBJECT IDENTIFIER ::= { snmpTlstmObjects 1 }
snmpTlstmSessionOpens    OBJECT-TYPE
SYNTAX                    Counter32
MAX-ACCESS                read-only
STATUS                    current
DESCRIPTION               "The number of times an openSession() request has been executed
```

```
    as a (D)TLS client, regardless of whether it succeeded or
    failed."
 ::= { snmpTlstmSession 1 }
snmpTlstmSessionClientCloses OBJECT-TYPE
SYNTAX      Counter32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "The number of times a closeSession() request has been
    executed as a (D)TLS client, regardless of whether it
    succeeded or failed."
 ::= { snmpTlstmSession 2 }
snmpTlstmSessionOpenErrors OBJECT-TYPE
SYNTAX      Counter32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "The number of times an openSession() request failed to open a
    session as a (D)TLS client, for any reason."
 ::= { snmpTlstmSession 3 }
snmpTlstmSessionAccepts OBJECT-TYPE
SYNTAX      Counter32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "The number of times a (D)TLS server has accepted a new
    connection from a client and has received at least one SNMP
    message through it."
 ::= { snmpTlstmSession 4 }

snmpTlstmSessionServerCloses OBJECT-TYPE
SYNTAX      Counter32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "The number of times a closeSession() request has been
    executed as a (D)TLS server, regardless of whether it
    succeeded or failed."
 ::= { snmpTlstmSession 5 }
snmpTlstmSessionNoSessions OBJECT-TYPE
SYNTAX      Counter32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "The number of times an outgoing message was dropped because
    the session associated with the passed tmStateReference was no
    longer (or was never) available."
 ::= { snmpTlstmSession 6 }
```



```
snmpTlstmSessionInvalidClientCertificates OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The number of times an incoming session was not established
        on a (D)TLS server because the presented client certificate
        was invalid. Reasons for invalidation include, but are not
        limited to, cryptographic validation failures or lack of a
        suitable mapping row in the snmpTlstmCertToTSNTable or the
        snmpTlstmCertToTSN13Table."
    ::= { snmpTlstmSession 7 }
snmpTlstmSessionUnknownServerCertificate OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The number of times an outgoing session was not established
        on a (D)TLS client because the server certificate presented
        by an SNMP over (D)TLS server was invalid because no
        configured fingerprint or Certification Authority (CA) was
        acceptable to validate it.
        This may result because there was no entry in the
        snmpTlstmAddrTable (or snmpTlstmAddr13Table) or because no
        path could be found to a known CA."
    ::= { snmpTlstmSession 8 }
snmpTlstmSessionInvalidServerCertificates OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The number of times an outgoing session was not established
        on a (D)TLS client because the server certificate presented
        by an SNMP over (D)TLS server could not be validated even if
        the fingerprint or expected validation path was known. That
        is, a cryptographic validation error occurred during
        certificate validation processing.
        Reasons for invalidation include, but are not
        limited to, cryptographic validation failures."
    ::= { snmpTlstmSession 9 }
snmpTlstmSessionInvalidCaches OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The number of outgoing messages dropped because the
        tmStateReference referred to an invalid cache."
    ::= { snmpTlstmSession 10 }
```

```

-- Configuration Objects
snmpTlstmConfig          OBJECT IDENTIFIER ::= {snmpTlstmObjects 2}
-- Certificate mapping
snmpTlstmCertificateMapping OBJECT IDENTIFIER ::= {snmpTlstmConfig 1}
snmpTlstmCertToTSNCount OBJECT-TYPE
    SYNTAX          Gauge32
    MAX-ACCESS      read-only
    STATUS           deprecated
    DESCRIPTION
        "A count of the number of entries in the
        snmpTlstmCertToTSNTable."
    ::= { snmpTlstmCertificateMapping 1 }
snmpTlstmCertToTSNTableLastChanged OBJECT-TYPE
    SYNTAX          TimeStamp
    MAX-ACCESS      read-only
    STATUS           deprecated
    DESCRIPTION
        "The value of sysUpTime.0 when the snmpTlstmCertToTSNTable was
        last modified through any means, or 0 if it has not been
        modified since the command responder was started."
    ::= { snmpTlstmCertificateMapping 2 }
snmpTlstmCertToTSNTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF SnmpTlstmCertToTSNEntry
    MAX-ACCESS      not-accessible
    STATUS           deprecated
    DESCRIPTION
        "This table is used by a (D)TLS server to map the (D)TLS
        client's presented X.509 certificate to a tmSecurityName.
        On an incoming (D)TLS/SNMP connection, the client's presented
        certificate must either be validated based on an established
        trust anchor, or it must directly match a fingerprint in this
        table. This table does not provide any mechanisms for
        configuring the trust anchors; the transfer of any needed
        trusted certificates for path validation is expected to occur
        through an out-of-band transfer.
        Once the certificate has been found acceptable (either by path
        validation or directly matching a fingerprint in this table),
        this table is consulted to determine the appropriate
        tmSecurityName to identify with the remote connection. This
        is done by considering each active row from this table in
        prioritized order according to its snmpTlstmCertToTSNID value.
        Each row's snmpTlstmCertToTSNFingerprint value determines
        whether the row is a match for the incoming connection:
        1) If the row's snmpTlstmCertToTSNFingerprint value
           identifies the presented certificate, then consider the
           row as a successful match.
        2) If the row's snmpTlstmCertToTSNFingerprint value
           identifies a locally held copy of a trusted CA

```

certificate and that CA certificate was used to validate the path to the presented certificate, then consider the row as a successful match.

Once a matching row has been found, the `snmpTlstmCertToTSNMapType` value can be used to determine how the `tmSecurityName` to associate with the session should be determined. See the `snmpTlstmCertToTSNMapType` column's DESCRIPTION for details on determining the `tmSecurityName` value. If it is impossible to determine a `tmSecurityName` from the row's data combined with the data presented in the certificate, then additional rows MUST be searched looking for another potential match. If a resulting `tmSecurityName` mapped from a given row is not compatible with the needed requirements of a `tmSecurityName` (e.g., VACM imposes a 32-octet-maximum length and the certificate derived `securityName` could be longer), then it must be considered an invalid match and additional rows MUST be searched looking for another potential match.

If no matching and valid row can be found, the connection MUST be closed and SNMP messages MUST NOT be accepted over it.

Missing values of `snmpTlstmCertToTSNID` are acceptable and implementations should continue to the next highest numbered row. It is recommended that administrators skip index values to leave room for the insertion of future rows (for example, use values of 10 and 20 when creating initial rows).

Users are encouraged to make use of certificates with `subjectAltName` fields that can be used as `tmSecurityNames` so that a single root CA certificate can allow all child certificate's `subjectAltName` to map directly to a `tmSecurityName` via a 1:1 transformation. However, this table is flexible to allow for situations where existing deployed certificate infrastructures do not provide adequate `subjectAltName` values for use as `tmSecurityNames`.

Direct mapping from each individual certificate fingerprint to a `tmSecurityName` is also possible but requires one entry in the table per `tmSecurityName` and requires more management operations to completely configure a device.

This table and its associated objects were deprecated because the fingerprint format changed to support TLS 1.3. By deprecating (and creating an updated) table, rather than just the fingerprint object, an implementation is able to support both the original TLS and new TLS 1.3 tables while forcing some agents to only use TLS 1.3."

```
 ::= { snmpTlstmCertificateMapping 3 }
snmpTlstmCertToTSNEntry OBJECT-TYPE
    SYNTAX      SnmpTlstmCertToTSNEntry
    MAX-ACCESS  not-accessible
```

```

STATUS      deprecated
DESCRIPTION
    "A row in the snmpTlstmCertToTSNTable that specifies a mapping
    for an incoming (D)TLS certificate to a tmSecurityName to use
    for a connection."
INDEX      { snmpTlstmCertToTSNID }
::= { snmpTlstmCertToTSNTable 1 }
snmpTlstmCertToTSNEntry ::= SEQUENCE {
    snmpTlstmCertToTSNID      Unsigned32,
    snmpTlstmCertToTSNFingerprint  SnmpTLSFingerprint,
    snmpTlstmCertToTSNMapType  AutonomousType,
    snmpTlstmCertToTSNData      OCTET STRING,
    snmpTlstmCertToTSNStorageType  StorageType,
    snmpTlstmCertToTSNRowStatus  RowStatus
}
snmpTlstmCertToTSNID OBJECT-TYPE
SYNTAX      Unsigned32 (1..4294967295)
MAX-ACCESS  not-accessible
STATUS      deprecated
DESCRIPTION
    "A unique, prioritized index for the given entry.  Lower
    numbers indicate a higher priority."
::= { snmpTlstmCertToTSNEntry 1 }
snmpTlstmCertToTSNFingerprint OBJECT-TYPE
SYNTAX      SnmpTLSFingerprint (SIZE(1..255))
MAX-ACCESS  read-create
STATUS      deprecated
DESCRIPTION
    "A cryptographic hash of an X.509 certificate.  The results of
    a successful matching fingerprint to either the trusted CA in
    the certificate validation path or to the certificate itself
    is dictated by the snmpTlstmCertToTSNMapType column.
    This object was deprecated because TLS 1.3 uses a 2-octet
    cipher suite identifier rather than a 1-octet hashing algorithm
    identifier."
::= { snmpTlstmCertToTSNEntry 2 }
snmpTlstmCertToTSNMapType OBJECT-TYPE
SYNTAX      AutonomousType
MAX-ACCESS  read-create
STATUS      deprecated
DESCRIPTION
    "Specifies the mapping type for deriving a tmSecurityName from
    a certificate.  Details for mapping of a particular type SHALL
    be specified in the DESCRIPTION clause of the OBJECT-IDENTITY
    that describes the mapping.  If a mapping succeeds it will
    return a tmSecurityName for use by the TLSTM model and
    processing stops.
    If the resulting mapped value is not compatible with the

```

needed requirements of a tmSecurityName (e.g., VACM imposes a 32-octet-maximum length and the certificate derived securityName could be longer), then future rows MUST be searched for additional snmpTlstmCertToTSNFingerprint matches to look for a mapping that succeeds. Suitable values for assigning to this object that are defined within the SNMP-TLS-TM-MIB can be found in the snmpTlstmCertToTSNMIdentities portion of the MIB tree."

```

DEFVAL { snmpTlstmCertSpecified }
::= { snmpTlstmCertToTSNEntry 3 }
snmpTlstmCertToTSNData OBJECT-TYPE
    SYNTAX      OCTET STRING (SIZE(0..1024))
    MAX-ACCESS   read-create
    STATUS       deprecated
    DESCRIPTION
        "Auxiliary data used as optional configuration information for
        a given mapping specified by the snmpTlstmCertToTSNMapType
        column. Only some mapping systems will make use of this
        column. The value in this column MUST be ignored for any
        mapping type that does not require data present in this
        column."
    DEFVAL { "" }
    ::= { snmpTlstmCertToTSNEntry 4 }
snmpTlstmCertToTSNStorageType OBJECT-TYPE
    SYNTAX      StorageType
    MAX-ACCESS   read-create
    STATUS       deprecated
    DESCRIPTION
        "The storage type for this conceptual row. Conceptual rows
        having the value 'permanent' need not allow write-access to
        any columnar objects in the row."
    DEFVAL      { nonVolatile }
    ::= { snmpTlstmCertToTSNEntry 5 }
snmpTlstmCertToTSNRowStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS   read-create
    STATUS       deprecated
    DESCRIPTION
        "The status of this conceptual row. This object may be used
        to create or remove rows from this table.
        To create a row in this table, an administrator must set this
        object to either createAndGo(4) or createAndWait(5).
        Until instances of all corresponding columns are appropriately
        configured, the value of the corresponding instance of the
        snmpTlstmParamsRowStatus column is notReady(3).
        In particular, a newly created row cannot be made active until
        the corresponding snmpTlstmCertToTSNFingerprint,
        snmpTlstmCertToTSNMapType, and snmpTlstmCertToTSNData columns

```

```

have been set.
The following objects may not be modified while the
value of this object is active(1):
    - snmpTlstmCertToTSNFingerprint
    - snmpTlstmCertToTSNMapType
    - snmpTlstmCertToTSNData
An attempt to set these objects while the value of
snmpTlstmParamsRowStatus is active(1) will result in
an inconsistentValue error."
 ::= { snmpTlstmCertToTSNEntry 6 }
-- Maps tmSecurityNames to certificates for use by SNMP-TARGET-MIB
snmpTlstmParamsCount OBJECT-TYPE
    SYNTAX      Gauge32
    MAX-ACCESS  read-only
    STATUS      deprecated
    DESCRIPTION
        "A count of the number of entries in the snmpTlstmParamsTable."
 ::= { snmpTlstmCertificateMapping 4 }
snmpTlstmParamsTableLastChanged OBJECT-TYPE
    SYNTAX      TimeStamp
    MAX-ACCESS  read-only
    STATUS      deprecated
    DESCRIPTION
        "The value of sysUpTime.0 when the snmpTlstmParamsTable
        was last modified through any means, or 0 if it has not been
        modified since the command responder was started."
 ::= { snmpTlstmCertificateMapping 5 }
snmpTlstmParamsTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF SnmpTlstmParamsEntry
    MAX-ACCESS  not-accessible
    STATUS      deprecated
    DESCRIPTION
        "This table is used by a (D)TLS client when a (D)TLS
        connection is being set up using an entry in the
        SNMP-TARGET-MIB. It extends the SNMP-TARGET-MIB's
        snmpTargetParamsTable with a fingerprint of a certificate to
        use when establishing such a (D)TLS connection."
 ::= { snmpTlstmCertificateMapping 6 }
snmpTlstmParamsEntry OBJECT-TYPE
    SYNTAX      SnmpTlstmParamsEntry
    MAX-ACCESS  not-accessible
    STATUS      deprecated
    DESCRIPTION
        "A conceptual row containing a fingerprint hash of a locally
        held certificate for a given snmpTargetParamsEntry. The
        values in this row should be ignored if the connection that
        needs to be established, as indicated by the SNMP-TARGET-MIB
        infrastructure, is not a certificate and TLS based

```

connection. The connection SHOULD NOT be established if the certificate fingerprint stored in this entry does not point to a valid locally held certificate or if it points to an unusable certificate (such as might happen when the certificate's expiration date has been reached)."

```

INDEX      { IMPLIED snmpTargetParamsName }
::= { snmpTlstmParamsTable 1 }
snmpTlstmParamsEntry ::= SEQUENCE {
    snmpTlstmParamsClientFingerprint SnmpTLSEFingerprint,
    snmpTlstmParamsStorageType        StorageType,
    snmpTlstmParamsRowStatus          RowStatus
}
snmpTlstmParamsClientFingerprint OBJECT-TYPE
SYNTAX      SnmpTLSEFingerprint
MAX-ACCESS  read-create
STATUS      deprecated
DESCRIPTION
    "This object stores the hash of the public portion of a
    locally held X.509 certificate. The X.509 certificate, its
    public key, and the corresponding private key will be used
    when initiating a TLS connection as a TLS client."
    ::= { snmpTlstmParamsEntry 1 }
snmpTlstmParamsStorageType OBJECT-TYPE
SYNTAX      StorageType
MAX-ACCESS  read-create
STATUS      deprecated
DESCRIPTION
    "The storage type for this conceptual row. Conceptual rows
    having the value 'permanent' need not allow write-access to
    any columnar objects in the row."
    DEFVAL   { nonVolatile }
    ::= { snmpTlstmParamsEntry 2 }
snmpTlstmParamsRowStatus OBJECT-TYPE
SYNTAX      RowStatus
MAX-ACCESS  read-create
STATUS      deprecated
DESCRIPTION
    "The status of this conceptual row. This object may be used
    to create or remove rows from this table.
    To create a row in this table, an administrator must set this
    object to either createAndGo(4) or createAndWait(5).
    Until instances of all corresponding columns are appropriately
    configured, the value of the corresponding instance of the
    snmpTlstmParamsRowStatus column is notReady(3).
    In particular, a newly created row cannot be made active until
    the corresponding snmpTlstmParamsClientFingerprint column has
    been set.
    The snmpTlstmParamsClientFingerprint object may not be modified

```

```
while the value of this object is active(1).
An attempt to set these objects while the value of
snmpTlstmParamsRowStatus is active(1) will result in
an inconsistentValue error."
 ::= { snmpTlstmParamsEntry 3 }
mpTlstmAddrCount OBJECT-TYPE
    SYNTAX      Gauge32
    MAX-ACCESS  read-only
    STATUS      deprecated
    DESCRIPTION
        "A count of the number of entries in the snmpTlstmAddrTable."
 ::= { snmpTlstmCertificateMapping 7 }
snmpTlstmAddrTableLastChanged OBJECT-TYPE
    SYNTAX      TimeStamp
    MAX-ACCESS  read-only
    STATUS      deprecated
    DESCRIPTION
        "The value of sysUpTime.0 when the snmpTlstmAddrTable
        was last modified through any means, or 0 if it has not been
        modified since the command responder was started."
 ::= { snmpTlstmCertificateMapping 8 }
snmpTlstmAddrTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF SnmpTlstmAddrEntry
    MAX-ACCESS  not-accessible
    STATUS      deprecated
    DESCRIPTION
        "This table is used by a TLS client when a TLS
        connection is being set up using an entry in the
        SNMP-TARGET-MIB. It extends the SNMP-TARGET-MIB's
        snmpTargetAddrTable so that the client can verify that the
        correct server has been reached. This verification can use
        either a certificate fingerprint, or an identity
        authenticated via certification path validation.
        If there is an active row in this table corresponding to the
        entry in the SNMP-TARGET-MIB that was used to establish the
        connection, and the row's snmpTlstmAddrServerFingerprint
        column has non-empty value, then the server's presented
        certificate is compared with the
        snmpTlstmAddrServerFingerprint value (and the
        snmpTlstmAddrServerIdentity column is ignored). If the
        fingerprint matches, the verification has succeeded. If the
        fingerprint does not match, then the connection MUST be
        closed.
        If the server's presented certificate has passed
        certification path validation [RFC5280] to a configured
        trust anchor, and an active row exists with a zero-length
        snmpTlstmAddrServerFingerprint value, then the
        snmpTlstmAddrServerIdentity column contains the expected
```


host name. This expected host name is then compared against the server's certificate as follows:

- Implementations MUST support matching the expected host name against a `dnsName` in the `subjectAltName` extension field
- The '*' (ASCII 0x2a) wildcard character is allowed in the `dnsName` of the `subjectAltName` extension, but only as the left-most (least significant) DNS label in that value. This wildcard matches any left-most DNS label in the server name. That is, the subject `*.example.com` matches the server names `a.example.com` and `b.example.com`, but does not match `example.com` or `a.b.example.com`. Implementations MUST support wildcards in certificates as specified above, but MAY provide a configuration option to disable them.
- If the locally configured name is an internationalized domain name, conforming implementations MUST convert it to the ASCII Compatible Encoding (ACE) format for performing comparisons, as specified in Section 7 of [RFC5280].

If the expected host name fails these conditions then the connection MUST be closed.

If there is no row in this table corresponding to the entry in the SNMP-TARGET-MIB and the server can be authorized by another, implementation-dependent means, then the connection MAY still proceed."

```
 ::= { snmpTlstmCertificateMapping 9 }
snmpTlstmAddrEntry OBJECT-TYPE
    SYNTAX      SnmpTlstmAddrEntry
    MAX-ACCESS  not-accessible
    STATUS      deprecated
    DESCRIPTION
        "A conceptual row containing a copy of a certificate's
        fingerprint for a given snmpTargetAddrEntry. The values in
        this row should be ignored if the connection that needs to be
        established, as indicated by the SNMP-TARGET-MIB
        infrastructure, is not a TLS based connection. If an
        snmpTlstmAddrEntry exists for a given snmpTargetAddrEntry, then
        the presented server certificate MUST match or the connection
        MUST NOT be established. If a row in this table does not
        exist to match an snmpTargetAddrEntry row, then the connection
        SHOULD still proceed if some other certificate validation path
        algorithm (e.g., RFC 5280) can be used."
    INDEX       { IMPLIED snmpTargetAddrName }
 ::= { snmpTlstmAddrTable 1 }
SnmpTlstmAddrEntry ::= SEQUENCE {
    snmpTlstmAddrServerFingerprint    SnmpTLSPFingerprint,
    snmpTlstmAddrServerIdentity       SnmpAdminString,
    snmpTlstmAddrStorageType          StorageType,
```

```
    snmpTlstmAddrRowStatus      RowStatus
}
snmpTlstmAddrServerFingerprint OBJECT-TYPE
    SYNTAX      SnmpTLSFingerprint
    MAX-ACCESS   read-create
    STATUS      deprecated
    DESCRIPTION
        "A cryptographic hash of a public X.509 certificate. This
        object should store the hash of the public X.509 certificate
        that the remote server should present during the TLS
        connection setup. The fingerprint of the presented
        certificate and this hash value MUST match exactly or the
        connection MUST NOT be established."
    DEFVAL { "" }
    ::= { snmpTlstmAddrEntry 1 }
snmpTlstmAddrServerIdentity OBJECT-TYPE
    SYNTAX      SnmpAdminString
    MAX-ACCESS   read-create
    STATUS      deprecated
    DESCRIPTION
        "The reference identity to check against the identity
        presented by the remote system."
    DEFVAL { "" }
    ::= { snmpTlstmAddrEntry 2 }
snmpTlstmAddrStorageType OBJECT-TYPE
    SYNTAX      StorageType
    MAX-ACCESS   read-create
    STATUS      deprecated
    DESCRIPTION
        "The storage type for this conceptual row. Conceptual rows
        having the value 'permanent' need not allow write-access to
        any columnar objects in the row."
    DEFVAL      { nonVolatile }
    ::= { snmpTlstmAddrEntry 3 }
snmpTlstmAddrRowStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS   read-create
    STATUS      deprecated
    DESCRIPTION
        "The status of this conceptual row. This object may be used
        to create or remove rows from this table.
        To create a row in this table, an administrator must set this
        object to either createAndGo(4) or createAndWait(5).
        Until instances of all corresponding columns are
        appropriately configured, the value of the
        corresponding instance of the snmpTlstmAddrRowStatus
        column is notReady(3).
        In particular, a newly created row cannot be made active until
```

the corresponding snmpTlstmAddrServerFingerprint column has been set.

Rows MUST NOT be active if the snmpTlstmAddrServerFingerprint column is blank and the snmpTlstmAddrServerIdentity is set to '*' since this would insecurely accept any presented certificate.

The snmpTlstmAddrServerFingerprint object may not be modified while the value of this object is active(1).

An attempt to set these objects while the value of snmpTlstmAddrRowStatus is active(1) will result in an inconsistentValue error."

```
 ::= { snmpTlstmAddrEntry 4 }
snmpTlstmCertToTSN13Count OBJECT-TYPE
    SYNTAX      Gauge32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "A count of the number of entries in the
        snmpTlstmCertToTSN13Table."
 ::= { snmpTlstmCertificateMapping 10 }
snmpTlstmCertToTSN13TableLastChanged OBJECT-TYPE
    SYNTAX      TimeStamp
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The value of sysUpTime.0 when the snmpTlstmCertToTSN13Table
        was last modified through any means, or 0 if it has not been
        modified since the command responder was started."
 ::= { snmpTlstmCertificateMapping 11 }
snmpTlstmCertToTSN13Table OBJECT-TYPE
    SYNTAX      SEQUENCE OF SnmpTlstmCertToTSN13Entry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This table is used by a TLS 1.3 server to map the TLS 1.3
        client's presented X.509 certificate to a tmSecurityName.
        On an incoming TLS/SNMP connection, the client's presented
        certificate must either be validated based on an established
        trust anchor, or it must directly match a fingerprint in this
        table. This table does not provide any mechanisms for
        configuring the trust anchors; the transfer of any needed
        trusted certificates for path validation is expected to occur
        through an out-of-band transfer.
        Once the certificate has been found acceptable (either by path
        validation or directly matching a fingerprint in this table),
        this table is consulted to determine the appropriate
        tmSecurityName to identify with the remote connection. This
```

is done by considering each active row from this table in prioritized order according to its `snmpTlstmCertToTSN13ID` value. Each row's `snmpTlstmCertToTSN13Fingerprint` value determines whether the row is a match for the incoming connection:

- 1) If the row's `snmpTlstmCertToTSN13Fingerprint` value identifies the presented certificate, then consider the row as a successful match.
- 2) If the row's `snmpTlstmCertToTSN13Fingerprint` value identifies a locally held copy of a trusted CA certificate and that CA certificate was used to validate the path to the presented certificate, then consider the row as a successful match.

Once a matching row has been found, the `snmpTlstmCertToTSN13MapType` value can be used to determine how the `tmSecurityName` to associate with the session should be determined. See the `snmpTlstmCertToTSN13MapType` column's DESCRIPTION for details on determining the `tmSecurityName` value. If it is impossible to determine a `tmSecurityName` from the row's data combined with the data presented in the certificate, then additional rows MUST be searched looking for another potential match. If a resulting `tmSecurityName` mapped from a given row is not compatible with the needed requirements of a `tmSecurityName` (e.g., VACM imposes a 32-octet-maximum length and the certificate derived `securityName` could be longer), then it must be considered an invalid match and additional rows MUST be searched looking for another potential match.

If no matching and valid row can be found, the connection MUST be closed and SNMP messages MUST NOT be accepted over it.

Missing values of `snmpTlstmCertToTSN13ID` are acceptable and implementations should continue to the next highest numbered row. It is recommended that administrators skip index values to leave room for the insertion of future rows (for example, use values of 10 and 20 when creating initial rows).

Users are encouraged to make use of certificates with `subjectAltName` fields that can be used as `tmSecurityNames` so that a single root CA certificate can allow all child certificate's `subjectAltName` to map directly to a `tmSecurityName` via a 1:1 transformation. However, this table is flexible to allow for situations where existing deployed certificate infrastructures do not provide adequate `subjectAltName` values for use as `tmSecurityNames`.

Direct mapping from each individual certificate fingerprint to a `tmSecurityName` is possible but requires one entry in the table per `tmSecurityName` and requires more management operations to completely configure a device."

```
::= { snmpTlstmCertificateMapping 12 }
```

```

snmpTlstmCertToTSN13Entry OBJECT-TYPE
    SYNTAX      SnmpTlstmCertToTSN13Entry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A row in the snmpTlstmCertToTSN13Table that specifies a
        mapping for an incoming TLS certificate to a tmSecurityName
        to use for a connection."
    INDEX       { snmpTlstmCertToTSN13ID }
    ::= { snmpTlstmCertToTSN13Table 1 }
SnmpTlstmCertToTSN13Entry ::= SEQUENCE {
    snmpTlstmCertToTSN13ID      Unsigned32,
    snmpTlstmCertToTSN13Fingerprint  SnmpTLS13Fingerprint,
    snmpTlstmCertToTSN13MapType   AutonomousType,
    snmpTlstmCertToTSN13Data      OCTET STRING,
    snmpTlstmCertToTSN13StorageType  StorageType,
    snmpTlstmCertToTSN13RowStatus   RowStatus
}
snmpTlstmCertToTSN13ID OBJECT-TYPE
    SYNTAX      Unsigned32 (1..4294967295)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A unique, prioritized index for the given entry.  Lower
        numbers indicate a higher priority."
    ::= { snmpTlstmCertToTSN13Entry 1 }
snmpTlstmCertToTSN13Fingerprint OBJECT-TYPE
    SYNTAX      SnmpTLS13Fingerprint (SIZE(2..255))
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "A cryptographic hash of an X.509 certificate.  The results of
        a successful matching fingerprint to either the trusted CA in
        the certificate validation path or to the certificate itself
        is dictated by the snmpTlstmCertToTSN13MapType column."
    ::= { snmpTlstmCertToTSN13Entry 2 }
snmpTlstmCertToTSN13MapType OBJECT-TYPE
    SYNTAX      AutonomousType
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "Specifies the mapping type for deriving a tmSecurityName from
        a certificate.  Details for mapping of a particular type SHALL
        be specified in the DESCRIPTION clause of the OBJECT-IDENTITY
        that describes the mapping.  If a mapping succeeds it will
        return a tmSecurityName for use by the TLSTM model and
        processing stops.
        If the resulting mapped value is not compatible with the

```

needed requirements of a tmSecurityName (e.g., VACM imposes a 32-octet-maximum length and the certificate derived securityName could be longer), then future rows MUST be searched for additional snmpTlstmCertToTSN13Fingerprint matches to look for a mapping that succeeds. Suitable values for assigning to this object that are defined within the SNMP-TLS-TM-MIB can be found in the snmpTlstmCertToTSNMIdentities portion of the MIB tree."

```

DEFVAL { snmpTlstmCertSpecified }
::= { snmpTlstmCertToTSN13Entry 3 }
snmpTlstmCertToTSN13Data OBJECT-TYPE
    SYNTAX      OCTET STRING (SIZE(0..1024))
    MAX-ACCESS   read-create
    STATUS       current
    DESCRIPTION
        "Auxiliary data used as optional configuration information for
        a given mapping specified by the snmpTlstmCertToTSN13MapType
        column. Only some mapping systems will make use of this
        column. The value in this column MUST be ignored for any
        mapping type that does not require data present in this
        column."
    DEFVAL { "" }
    ::= { snmpTlstmCertToTSN13Entry 4 }
snmpTlstmCertToTSN13StorageType OBJECT-TYPE
    SYNTAX      StorageType
    MAX-ACCESS   read-create
    STATUS       current
    DESCRIPTION
        "The storage type for this conceptual row. Conceptual rows
        having the value 'permanent' need not allow write-access to
        any columnar objects in the row."
    DEFVAL      { nonVolatile }
    ::= { snmpTlstmCertToTSN13Entry 5 }
snmpTlstmCertToTSN13RowStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS   read-create
    STATUS       current
    DESCRIPTION
        "The status of this conceptual row. This object may be used
        to create or remove rows from this table.
        To create a row in this table, an administrator must set this
        object to either createAndGo(4) or createAndWait(5).
        Until instances of all corresponding columns are appropriately
        configured, the value of the corresponding instance of the
        snmpTlstmParams13RowStatus column is notReady(3).
        In particular, a newly created row cannot be made active until
        the corresponding snmpTlstmCertToTSN13Fingerprint,
        snmpTlstmCertToTSN13MapType, and snmpTlstmCertToTSN13Data

```

columns have been set.
 The following objects may not be modified while the value of this object is active(1):

- snmpTlstmCertToTSN13Fingerprint
- snmpTlstmCertToTSN13MapType
- snmpTlstmCertToTSN13Data

An attempt to set these objects while the value of snmpTlstmParams13RowStatus is active(1) will result in an inconsistentValue error."

```
 ::= { snmpTlstmCertToTSN13Entry 6 }
snmpTlstmParams13Count OBJECT-TYPE
    SYNTAX      Gauge32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "A count of the number of entries in the
        snmpTlstmParams13Table."
 ::= { snmpTlstmCertificateMapping 13 }
snmpTlstmParams13TableLastChanged OBJECT-TYPE
    SYNTAX      TimeStamp
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The value of sysUpTime.0 when the snmpTlstmParams13Table
        was last modified through any means, or 0 if it has not been
        modified since the command responder was started."
 ::= { snmpTlstmCertificateMapping 14 }
snmpTlstmParams13Table OBJECT-TYPE
    SYNTAX      SEQUENCE OF SnmpTlstmParams13Entry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This table is used by a TLS client when a TLS
        connection is being set up using an entry in the
        SNMP-TARGET-MIB. It extends the SNMP-TARGET-MIB's
        snmpTargetParams13Table with a fingerprint of a certificate to
        use when establishing such a TLS connection."
 ::= { snmpTlstmCertificateMapping 15 }
snmpTlstmParams13Entry OBJECT-TYPE
    SYNTAX      SnmpTlstmParams13Entry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A conceptual row containing a fingerprint hash of a locally
        held certificate for a given snmpTargetParamsEntry. The
        values in this row should be ignored if the connection that
        needs to be established, as indicated by the SNMP-TARGET-MIB
        infrastructure, is not a certificate and TLS based
```

connection. The connection SHOULD NOT be established if the certificate fingerprint stored in this entry does not point to a valid locally held certificate or if it points to an unusable certificate (such as might happen when the certificate's expiration date has been reached)."

```

INDEX      { IMPLIED snmpTargetParamsName }
::= { snmpTlstmParams13Table 1 }
snmpTlstmParams13Entry ::= SEQUENCE {
    snmpTlstmParams13ClientFingerprint SnmpTLS13Fingerprint,
    snmpTlstmParams13StorageType        StorageType,
    snmpTlstmParams13RowStatus          RowStatus
}
snmpTlstmParams13ClientFingerprint OBJECT-TYPE
SYNTAX      SnmpTLS13Fingerprint
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
    "This object stores the hash of the public portion of a
    locally held X.509 certificate. The X.509 certificate, its
    public key, and the corresponding private key will be used
    when initiating a TLS connection as a TLS client."
    ::= { snmpTlstmParams13Entry 1 }
snmpTlstmParams13StorageType OBJECT-TYPE
SYNTAX      StorageType
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
    "The storage type for this conceptual row. Conceptual rows
    having the value 'permanent' need not allow write-access to
    any columnar objects in the row."
    DEFVAL   { nonVolatile }
    ::= { snmpTlstmParams13Entry 2 }
snmpTlstmParams13RowStatus OBJECT-TYPE
SYNTAX      RowStatus
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
    "The status of this conceptual row. This object may be used
    to create or remove rows from this table.
    To create a row in this table, an administrator must set this
    object to either createAndGo(4) or createAndWait(5).
    Until instances of all corresponding columns are appropriately
    configured, the value of the corresponding instance of the
    snmpTlstmParams13RowStatus column is notReady(3).
    In particular, a newly created row cannot be made active until
    the corresponding snmpTlstmParams13ClientFingerprint column has
    been set.
    The snmpTlstmParams13ClientFingerprint object may not be

```


modified while the value of this object is active(1).
An attempt to set these objects while the value of
snmpTlstmParams13RowStatus is active(1) will result in
an inconsistentValue error."
 ::= { snmpTlstmParams13Entry 3 }
snmpTlstmAddr13Count OBJECT-TYPE
SYNTAX Gauge32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
 "A count of the number of entries in the snmpTlstmAddr13Table."
 ::= { snmpTlstmCertificateMapping 16 }
snmpTlstmAddr13TableLastChanged OBJECT-TYPE
SYNTAX TimeStamp
MAX-ACCESS read-only
STATUS current
DESCRIPTION
 "The value of sysUpTime.0 when the snmpTlstmAddr13Table
 was last modified through any means, or 0 if it has not been
 modified since the command responder was started."
 ::= { snmpTlstmCertificateMapping 17 }
snmpTlstmAddr13Table OBJECT-TYPE
SYNTAX SEQUENCE OF SnmpTlstmAddr13Entry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
 "This table is used by a TLS client when a TLS
 connection is being set up using an entry in the
 SNMP-TARGET-MIB. It extends the SNMP-TARGET-MIB's
 snmpTargetAddrTable so that the client can verify that the
 correct server has been reached. This verification can use
 either a certificate fingerprint, or an identity
 authenticated via certification path validation.
 If there is an active row in this table corresponding to the
 entry in the SNMP-TARGET-MIB that was used to establish the
 connection, and the row's snmpTlstmAddr13ServerFingerprint
 column has non-empty value, then the server's presented
 certificate is compared with the
 snmpTlstmAddr13ServerFingerprint value (and the
 snmpTlstmAddr13ServerIdentity column is ignored). If the
 fingerprint matches, the verification has succeeded. If the
 fingerprint does not match, then the connection MUST be
 closed.
 If the server's presented certificate has passed
 certification path validation [RFC5280] to a configured
 trust anchor, and an active row exists with a zero-length
 snmpTlstmAddr13ServerFingerprint value, then the
 snmpTlstmAddr13ServerIdentity column contains the expected

host name. This expected host name is then compared against the server's certificate as follows:

- Implementations MUST support matching the expected host name against a `dnsName` in the `subjectAltName` extension field.
- The '*' (ASCII 0x2a) wildcard character is allowed in the `dnsName` of the `subjectAltName` extension, but only as the left-most (least significant) DNS label in that value. This wildcard matches any left-most DNS label in the server name. That is, the subject `*.example.com` matches the server names `a.example.com` and `b.example.com`, but does not match `example.com` or `a.b.example.com`. Implementations MUST support wildcards in certificates as specified above, but MAY provide a configuration option to disable them.
- If the locally configured name is an internationalized domain name, conforming implementations MUST convert it to the ASCII Compatible Encoding (ACE) format for performing comparisons, as specified in Section 7 of [RFC5280].

If the expected host name fails these conditions then the connection MUST be closed.

If there is no row in this table corresponding to the entry in the SNMP-TARGET-MIB and the server can be authorized by another, implementation-dependent means, then the connection MAY still proceed."

```
 ::= { snmpTlstmCertificateMapping 18 }
snmpTlstmAddr13Entry OBJECT-TYPE
SYNTAX      SnmpTlstmAddr13Entry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "A conceptual row containing a copy of a certificate's
    fingerprint for a given snmpTargetAddrEntry. The values in
    this row should be ignored if the connection that needs to be
    established, as indicated by the SNMP-TARGET-MIB
    infrastructure, is not a TLS based connection. If an
    snmpTlstmAddr13Entry exists for a given snmpTargetAddrEntry,
    then the presented server certificate MUST match or the
    connection MUST NOT be established. If a row in this table
    does not exist to match an snmpTargetAddrEntry row, then the
    connection SHOULD still proceed if some other certificate
    validation path algorithm (e.g., RFC 5280) can be used."
INDEX       { IMPLIED snmpTargetAddrName }
 ::= { snmpTlstmAddr13Table 1 }
SnmpTlstmAddr13Entry ::= SEQUENCE {
    snmpTlstmAddr13ServerFingerprint    SnmpTLS13Fingerprint,
    snmpTlstmAddr13ServerIdentity        SnmpAdminString,
    snmpTlstmAddr13StorageType           StorageType,
```

```

        snmpTlstmAddr13RowStatus          RowStatus
    }
snmpTlstmAddr13ServerFingerprint OBJECT-TYPE
    SYNTAX          SnmpTLS13Fingerprint
    MAX-ACCESS      read-create
    STATUS           current
    DESCRIPTION
        "A cryptographic hash of a public X.509 certificate. This
        object should store the hash of the public X.509 certificate
        that the remote server should present during the TLS
        connection setup. The fingerprint of the presented
        certificate and this hash value MUST match exactly or the
        connection MUST NOT be established."
    DEFVAL { "" }
    ::= { snmpTlstmAddr13Entry 1 }
snmpTlstmAddr13ServerIdentity OBJECT-TYPE
    SYNTAX          SnmpAdminString
    MAX-ACCESS      read-create
    STATUS           current
    DESCRIPTION
        "The reference identity to check against the identity
        presented by the remote system."
    DEFVAL { "" }
    ::= { snmpTlstmAddr13Entry 2 }
snmpTlstmAddr13StorageType OBJECT-TYPE
    SYNTAX          StorageType
    MAX-ACCESS      read-create
    STATUS           current
    DESCRIPTION
        "The storage type for this conceptual row. Conceptual rows
        having the value 'permanent' need not allow write-access to
        any columnar objects in the row."
    DEFVAL          { nonVolatile }
    ::= { snmpTlstmAddr13Entry 3 }
snmpTlstmAddr13RowStatus OBJECT-TYPE
    SYNTAX          RowStatus
    MAX-ACCESS      read-create
    STATUS           current
    DESCRIPTION
        "The status of this conceptual row. This object may be used
        to create or remove rows from this table.
        To create a row in this table, an administrator must set this
        object to either createAndGo(4) or createAndWait(5).
        Until instances of all corresponding columns are
        appropriately configured, the value of the
        corresponding instance of the snmpTlstmAddr13RowStatus
        column is notReady(3).
        In particular, a newly created row cannot be made active until

```

the corresponding snmpTlstmAddr13ServerFingerprint column has been set.

Rows MUST NOT be active if the snmpTlstmAddr13ServerFingerprint column is blank and the snmpTlstmAddr13ServerIdentity is set to '*' since this would insecurely accept any presented certificate.

The snmpTlstmAddr13ServerFingerprint object may not be modified while the value of this object is active(1).

An attempt to set these objects while the value of snmpTlstmAddr13RowStatus is active(1) will result in an inconsistentValue error."

```

 ::= { snmpTlstmAddr13Entry 4 }
-- *****
-- snmpTlstmNotifications - Notifications Information
-- *****
snmpTlstmServerCertificateUnknown NOTIFICATION-TYPE
  OBJECTS { snmpTlstmSessionUnknownServerCertificate }
  STATUS current
  DESCRIPTION
    "Notification that the server certificate presented by an SNMP
    over (D)TLS server was invalid because no configured
    fingerprint or CA was acceptable to validate it. This may be
    because there was no entry in the snmpTlstmAddrTable (or
    snmpTlstmAddr13Table) or
    because no path could be found to known Certification
    Authority.
    To avoid notification loops, this notification MUST NOT be
    sent to servers that themselves have triggered the
    notification."
  ::= { snmpTlstmNotifications 1 }
snmpTlstmServerInvalidCertificate NOTIFICATION-TYPE
  OBJECTS { snmpTlstmAddrServerFingerprint,
            snmpTlstmSessionInvalidServerCertificates}
  STATUS deprecated
  DESCRIPTION
    "Notification that the server certificate presented by an SNMP
    over (D)TLS server could not be validated even if the
    fingerprint or expected validation path was known. That is, a
    cryptographic validation error occurred during certificate
    validation processing.
    To avoid notification loops, this notification MUST NOT be
    sent to servers that themselves have triggered the
    notification."
  ::= { snmpTlstmNotifications 2 }
snmpTlstmServerInvalidCertificate13 NOTIFICATION-TYPE
  OBJECTS { snmpTlstmAddr13ServerFingerprint,
            snmpTlstmSessionInvalidServerCertificates}
  STATUS current

```

DESCRIPTION

"Notification that the server certificate presented by an SNMP over TLS server could not be validated even if the fingerprint or expected validation path was known. That is, a cryptographic validation error occurred during certificate validation processing.

To avoid notification loops, this notification MUST NOT be sent to servers that themselves have triggered the notification."

```
 ::= { snmpTlstmNotifications 3 }
-- *****
-- snmpTlstmCompliances - Conformance Information
-- *****
snmpTlstmCompliances OBJECT IDENTIFIER ::= { snmpTlstmConformance 1 }
snmpTlstmGroups OBJECT IDENTIFIER ::= { snmpTlstmConformance 2 }
-- *****
-- Compliance statements
-- *****
snmpTlstmCompliance MODULE-COMPLIANCE
  STATUS      deprecated
  DESCRIPTION
    "The compliance statement for SNMP engines that support the
    SNMP-TLS-TM-MIB"
  MODULE
    MANDATORY-GROUPS { snmpTlstmStatsGroup,
                        snmpTlstmIncomingGroup,
                        snmpTlstmOutgoingGroup,
                        snmpTlstmNotificationGroup }
  ::= { snmpTlstmCompliances 1 }
snmpTlstmCompliance13 MODULE-COMPLIANCE
  STATUS      current
  DESCRIPTION
    "The compliance statement for SNMP engines that support the
    SNMP-TLS-TM-MIB"
  MODULE
    MANDATORY-GROUPS { snmpTlstmStatsGroup,
                        snmpTlstmIncoming13Group,
                        snmpTlstmOutgoing13Group,
                        snmpTlstmNotification13Group }
  ::= { snmpTlstmCompliances 2 }
-- *****
-- Units of conformance
-- *****
snmpTlstmStatsGroup OBJECT-GROUP
  OBJECTS {
    snmpTlstmSessionOpens,
    snmpTlstmSessionClientCloses,
    snmpTlstmSessionOpenErrors,
```

```
    snmpTlstmSessionAccepts,
    snmpTlstmSessionServerCloses,
    snmpTlstmSessionNoSessions,
    snmpTlstmSessionInvalidClientCertificates,
    snmpTlstmSessionUnknownServerCertificate,
    snmpTlstmSessionInvalidServerCertificates,
    snmpTlstmSessionInvalidCaches
}
STATUS      current
DESCRIPTION
    "A collection of objects for maintaining
    statistical information of an SNMP engine that
    implements the SNMP TLS Transport Model."
 ::= { snmpTlstmGroups 1 }
snmpTlstmIncomingGroup OBJECT-GROUP
OBJECTS {
    snmpTlstmCertToTSNCount,
    snmpTlstmCertToTSNTableLastChanged,
    snmpTlstmCertToTSNFingerprint,
    snmpTlstmCertToTSNMapType,
    snmpTlstmCertToTSNData,
    snmpTlstmCertToTSNStorageType,
    snmpTlstmCertToTSNRowStatus
}
STATUS      deprecated
DESCRIPTION
    "A collection of objects for maintaining
    incoming connection certificate mappings to
    tmSecurityNames of an SNMP engine that implements the
    SNMP TLS Transport Model."
 ::= { snmpTlstmGroups 2 }
snmpTlstmOutgoingGroup OBJECT-GROUP
OBJECTS {
    snmpTlstmParamsCount,
    snmpTlstmParamsTableLastChanged,
    snmpTlstmParamsClientFingerprint,
    snmpTlstmParamsStorageType,
    snmpTlstmParamsRowStatus,
    snmpTlstmAddrCount,
    snmpTlstmAddrTableLastChanged,
    snmpTlstmAddrServerFingerprint,
    snmpTlstmAddrServerIdentity,
    snmpTlstmAddrStorageType,
    snmpTlstmAddrRowStatus
}
STATUS      deprecated
DESCRIPTION
    "A collection of objects for maintaining
```

```
        outgoing connection certificates to use when opening
        connections as a result of SNMP-TARGET-MIB settings."
 ::= { snmpTlstmGroups 3 }
snmpTlstmNotificationGroup NOTIFICATION-GROUP
NOTIFICATIONS {
    snmpTlstmServerCertificateUnknown,
    snmpTlstmServerInvalidCertificate
}
STATUS deprecated
DESCRIPTION
    "Notifications"
 ::= { snmpTlstmGroups 4 }
snmpTlstmIncomingI3Group OBJECT-GROUP
OBJECTS {
    snmpTlstmCertToTSN13Count,
    snmpTlstmCertToTSN13TableLastChanged,
    snmpTlstmCertToTSN13Fingerprint,
    snmpTlstmCertToTSN13MapType,
    snmpTlstmCertToTSN13Data,
    snmpTlstmCertToTSN13StorageType,
    snmpTlstmCertToTSN13RowStatus
}
STATUS current
DESCRIPTION
    "A collection of objects for maintaining
    incoming connection certificate mappings to
    tmSecurityNames of an SNMP engine that implements the
    SNMP TLS 1.3 Transport Model."
 ::= { snmpTlstmGroups 5 }
snmpTlstmOutgoingI3Group OBJECT-GROUP
OBJECTS {
    snmpTlstmParamsI3Count,
    snmpTlstmParamsI3TableLastChanged,
    snmpTlstmParamsI3ClientFingerprint,
    snmpTlstmParamsI3StorageType,
    snmpTlstmParamsI3RowStatus,
    snmpTlstmAddrI3Count,
    snmpTlstmAddrI3TableLastChanged,
    snmpTlstmAddrI3ServerFingerprint,
    snmpTlstmAddrI3ServerIdentity,
    snmpTlstmAddrI3StorageType,
    snmpTlstmAddrI3RowStatus
}
STATUS current
DESCRIPTION
    "A collection of objects for maintaining
    outgoing connection certificates to use when opening
    TLS 1.3 connections as a result of SNMP-TARGET-MIB settings."
```

```
 ::= { snmpTlstmGroups 6 }
snmpTlstmNotification13Group NOTIFICATION-GROUP
NOTIFICATIONS {
    snmpTlstmServerCertificateUnknown,
    snmpTlstmServerInvalidCertificate13
}
STATUS current
DESCRIPTION
    "Notifications for the SNMP TLS 1.3 Transport Model"
 ::= { snmpTlstmGroups 7 }
END
```

5. Security Considerations

This document updates a transport model that permits SNMP to utilize TLS security services. The security threats and how the TLS transport model mitigates these threats are covered throughout this document and in [RFC6353]. Security considerations for TLS are described in Section 10 and Appendix E of TLS 1.3 [RFC8446].

5.1. MIB Module Security

There are a number of management objects defined in this MIB module with a MAX-ACCESS clause of read-write and/or read-create. Such objects might be considered sensitive or vulnerable in some network environments. The support for SET operations in a non-secure environment without proper protection can have a negative effect on network operations. These are the tables and objects and their sensitivity/vulnerability:

- * The `snmpTlstmParams13Table` can be used to change the outgoing X.509 certificate used to establish a TLS connection. Modifications to objects in this table need to be adequately authenticated since modifying the values in this table will have profound impacts to the security of outbound connections from the device. Since knowledge of authorization rules and certificate usage mechanisms might be considered sensitive, protection from disclosure of the SNMP traffic via encryption is automatically achieved via TLS 1.3.
- * The `snmpTlstmAddr13Table` can be used to change the expectations of the certificates presented by a remote TLS server. Modifications to objects in this table need to be adequately authenticated since modifying the values in this table will have profound impacts to the security of outbound connections from the device. Since knowledge of authorization rules and certificate usage mechanisms might be considered sensitive, protection from disclosure of the SNMP traffic via encryption is automatically achieved via TLS 1.3.

- * The `snmpTlstmCertToTSN13Table` is used to specify the mapping of incoming X.509 certificates to `tmSecurityNames`, which eventually get mapped to an SNMPv3 `securityName`. Modifications to objects in this table need to be adequately authenticated since modifying the values in this table will have profound impacts to the security of incoming connections to the device. Since knowledge of authorization rules and certificate usage mechanisms might be considered sensitive, protection from disclosure of the SNMP traffic via encryption is automatically achieved via TLS 1.3. When this table contains a significant number of rows it might affect the system performance when accepting new TLS connections.

Some of the readable objects in this MIB module (i.e., objects with a MAX-ACCESS other than not-accessible) might be considered sensitive or vulnerable in some network environments. It is thus important to control even GET and/or NOTIFY access to these objects and encrypt the values of these objects when sending them over the network via SNMP. These are the tables and objects and their sensitivity/vulnerability:

- * This MIB contains a collection of counters that monitor the TLS connections being established with a device. Since knowledge of connection and certificate usage mechanisms might be considered sensitive, protection from disclosure of the SNMP traffic via encryption is automatically achieved via TLS 1.3.

SNMP versions prior to SNMPv3 did not include adequate security. Even if the network itself is secure (for example, by using IPsec), even then, there is no control as to who on the secure network is allowed to access and GET/SET (read/change/create/delete) the objects in this MIB module.

As defined in Section 2.4, TLSTM clients and servers MUST NOT request, offer, or use SNMPv1 or SNMPv2c message processing described in [RFC3584], or the User-based Security Model of SNMPv3. Instead, it is RECOMMENDED to deploy SNMPv3 and to enable cryptographic security. It is then a customer/operator responsibility to ensure that the SNMP entity giving access to an instance of this MIB module is properly configured to give access to the objects only to those principals (users) that have legitimate rights to indeed GET or SET (change/create/delete) them.

6. IANA Considerations

This document has no IANA actions beyond those performed as a part of [RFC6353].

7. Acknowledgements

Acknowledgements This document is based on [RFC6353]. This document was reviewed by the following people who helped provide useful comments: Michaela Vanderveen.

8. References

8.1. Normative References

- [I-D.ietf-tls-dtls13]
Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-dtls13-43, 30 April 2021,
<<https://datatracker.ietf.org/doc/html/draft-ietf-tls-dtls13-43>>.
- [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>>.
- [RFC3584] Frye, R., Levi, D., Routhier, S., and B. Wijnen, "Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework", BCP 74, RFC 3584, DOI 10.17487/RFC3584, August 2003,
<<https://www.rfc-editor.org/info/rfc3584>>.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008,
<<https://www.rfc-editor.org/info/rfc5280>>.
- [RFC5591] Harrington, D. and W. Hardaker, "Transport Security Model for the Simple Network Management Protocol (SNMP)", STD 78, RFC 5591, DOI 10.17487/RFC5591, June 2009,
<<https://www.rfc-editor.org/info/rfc5591>>.
- [RFC6353] Hardaker, W., "Transport Layer Security (TLS) Transport Model for the Simple Network Management Protocol (SNMP)", STD 78, RFC 6353, DOI 10.17487/RFC6353, July 2011,
<<https://www.rfc-editor.org/info/rfc6353>>.
- [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>>.

- [STD62] Harrington, D., Presuhn, R., and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", STD 62, RFC 3411, December 2002.
- Case, J., Harrington, D., Presuhn, R., and B. Wijnen, "Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3412, December 2002.
- Levi, D., Meyer, P., and B. Stewart, "Simple Network Management Protocol (SNMP) Applications", STD 62, RFC 3413, December 2002.
- Blumenthal, U. and B. Wijnen, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)", STD 62, RFC 3414, December 2002.
- Wijnen, B., Presuhn, R., and K. McCloghrie, "View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3415, December 2002.
- Presuhn, R., Ed., "Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3416, December 2002.
- Presuhn, R., Ed., "Transport Mappings for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3417, December 2002.
- Presuhn, R., Ed., "Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3418, December 2002.

8.2. Informative References

- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<https://www.rfc-editor.org/info/rfc5246>>.
- [STD58] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.

McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Textual Conventions for SMIV2", STD 58, RFC 2579, April 1999.

McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Conformance Statements for SMIV2", STD 58, RFC 2580, April 1999.

Appendix A. Target and Notification Configuration Example

The following sections describe example configuration for the SNMP-TLS-TM-MIB, the SNMP-TARGET-MIB, the NOTIFICATION-MIB, and the SNMP-VIEW-BASED-ACM-MIB.

A.1. Configuring a Notification Originator

The following row adds the "Joe Cool" user to the "administrators" group:

```
vacmSecurityModel          = 4 (TSM)
vacmSecurityName           = "Joe Cool"
vacmGroupName              = "administrators"
vacmSecurityToGroupStorageType = 3 (nonVolatile)
vacmSecurityToGroupStatus   = 4 (createAndGo)
```

The following row configures the snmpTlstmAddr13Table to use certificate path validation and to require the remote notification receiver to present a certificate for the "server.example.org" identity.

```
snmpTargetAddrName        = "toNRAddr"
snmpTlstmAddr13ServerFingerprint = ""
snmpTlstmAddr13ServerIdentity = "server.example.org"
snmpTlstmAddr13StorageType = 3 (nonVolatile)
snmpTlstmAddr13RowStatus   = 4 (createAndGo)
```

The following row configures the snmpTargetAddrTable to send notifications using TLS/TCP to the snmpTls-trap port at 192.0.2.1:

```
snmpTargetAddrName        = "toNRAddr"
snmpTargetAddrTDomain     = snmpTLSTCPDomain
snmpTargetAddrTAddress     = "192.0.2.1:10162"
snmpTargetAddrTimeout     = 1500
snmpTargetAddrRetryCount  = 3
snmpTargetAddrTagList     = "toNRTag"
snmpTargetAddrParams      = "toNR" (MUST match below)
snmpTargetAddrStorageType = 3 (nonVolatile)
snmpTargetAddrRowStatus   = 4 (createAndGo)
```

The following row configures the `snmpTargetParamsTable` to send the notifications to "Joe Cool", using `authPriv` SNMPv3 notifications through the `TransportSecurityModel` [[RFC5591]]:

```

snmpTargetParamsName      = "toNR"      (MUST match above)
snmpTargetParamsMPModel   = 3 (SNMPv3)
snmpTargetParamsSecurityModel = 4 (TransportSecurityModel)
snmpTargetParamsSecurityName = "Joe Cool"
snmpTargetParamsSecurityLevel = 3          (authPriv)
snmpTargetParamsStorageType = 3          (nonVolatile)
snmpTargetParamsRowStatus  = 4          (createAndGo)

```

A.2. Configuring TLSTM to Utilize a Simple Derivation of `tmSecurityName`

The following row configures the `snmpTlstmCertToTSN13Table` to map a validated client certificate, referenced by the client's public X.509 hash fingerprint, to a `tmSecurityName` using the `subjectAltName` component of the certificate.

```

snmpTlstmCertToTSN13ID      = 1
                             (chosen by ordering preference)
snmpTlstmCertToTSN13Fingerprint = HASH (appropriate fingerprint)
snmpTlstmCertToTSN13MapType  = snmpTlstmCertSANAny
snmpTlstmCertToTSN13Data     = "" (not used)
snmpTlstmCertToTSN13StorageType = 3 (nonVolatile)
snmpTlstmCertToTSN13RowStatus = 4 (createAndGo)

```

This type of configuration should only be used when the naming conventions of the (possibly multiple) Certification Authorities are well understood, so two different principals cannot inadvertently be identified by the same derived `tmSecurityName`.

A.3. Configuring TLSTM to Utilize Table-Driven Certificate Mapping

The following row configures the `snmpTlstmCertToTSN13Table` to map a validated client certificate, referenced by the client's public X.509 hash fingerprint, to the directly specified `tmSecurityName` of "Joe Cool".

```

snmpTlstmCertToTSN13ID      = 2
                             (chosen by ordering preference)
snmpTlstmCertToTSN13Fingerprint = HASH (appropriate fingerprint)
snmpTlstmCertToTSN13MapType  = snmpTlstmCertSpecified
snmpTlstmCertToTSN13SecurityName = "Joe Cool"
snmpTlstmCertToTSN13StorageType = 3 (nonVolatile)
snmpTlstmCertToTSN13RowStatus = 4 (createAndGo)

```

Author's Address

Kenneth Vaughn (editor)
Trevilon LLC
6606 FM 1488 RD
Suite 148-503
Magnolia, TX 77354
United States of America

Phone: +1 571 331 5670
Email: kvaughn@trevilon.com

CCAMP Working Group
Internet-Draft
Intended status: Standards Track
Expires: 14 May 2022

C. Yu
I. Busi
Huawei Technologies
A. Guo
Futurewei Technologies
S. Belotti
Nokia
J-F. Bouquier
Vodafone
F. Peruzzini
TIM
O. Gonzalez de Dios
Telefonica
V. Lopez
Nokia
10 November 2021

A YANG Data Model for Optical Network Inventory
draft-yg3bp-ccamp-optical-inventory-yang-01

Abstract

This document defines a YANG data model for optical network inventory data information.

The YANG data model presented in this document is intended to be used as the basis toward a generic YANG data model for network inventory data information which can be augmented, when required, with technology-specific (e.g., optical) inventory data, to be defined either in a future version of this document or in another document.

The YANG data model defined in this document conforms to the Network Management Datastore Architecture (NMDA).

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 14 May 2022.

Copyright Notice

Copyright (c) 2021 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 Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Terminology and Notations	4
1.2. Tree Diagram	5
1.3. Prefix in Data Node Names	6
2. YANG Data Model for Optical Network Inventory	6
2.1. YANG Model Overview	6
3. Optical Network Inventory Tree Diagram	9
4. YANG Model for Optical Network Inventory	10
5. Manageability Considerations	16
6. Security Considerations	17
7. IANA Considerations	17
8. References	17
8.1. Normative References	17
8.2. Informative References	18
Acknowledgments	18
Authors' Addresses	18

1. Introduction

Network inventory management is a key component in operators' OSS architectures.

Network inventory is a fundamental functionality in network management and was specified many years ago. Given the emerging of data models and their deployment in operator's management and control systems, the traditional function of inventory management is also requested to be defined as a data model.

Network inventory management and monitoring is a critical part of ensuring the network stays healthy, well-planned, and functioning in the operator's network. Network inventory management allows the operator to keep track of what physical network devices are staying in the network including relevant software and hardware.

The network inventory management also helps the operator to know when to acquire new assets and what is needed, or to decommission old or faulty ones, which can help to improve network performance and capacity planning.

In [I-D.ietf-teas-actn-poi-applicability] a gap was identified regarding the lack of a YANG data model that could be used at ACTN MPI interface level to report whole/partial hardware inventory information available at PNC level towards north-bound systems (e.g., MDSC or OSS layer).

[RFC8345] initial goal was to make possible the augmentation of the YANG data model with network inventory data model but this was never developed and the scope was kept limited to network topology data only.

It is key for operators to drive the industry towards the use of a standard YANG data model for network inventory data instead of using vendors proprietary APIs (e.g., REST API).

In the ACTN architecture, this would bring also clear benefits at MDSC level for packet over optical integration scenarios since this would enable the correlation of the inventory information with the links information reported in the network topology model.

The intention is to define a generic YANG data model that would be as much as possible technology agnostic (valid for IP, optical and microwave networks) and that could be augmented, when required, to include some technology-specific inventory details.

[RFC8348] defines a YANG data model for the management of the hardware on a single server and therefore it is more applicable to the PNC South Bound Interface (SBI) towards the network elements rather than at the PNC MPI. However, the YANG data model defined in [RFC8348] has been used as a reference for defining the YANG network inventory data model.

For optical network inventory, the network inventory YANG data model should support the use cases (4a and 4b) and requirements defined in [ONF_TR-547], in order to guarantee a seamless integration at MDSC/OSS/orchestration layers.

The proposed YANG data model has been analysed to cover the requirements and use cases for Optical Network Inventory.

Being based on [RFC8348], this data model should be a good starting point toward a generic data model and applicable to any technology. However, further analysis of requirements and use cases is needed to extend the applicability of this YANG data model to other types of networks (IP and microwave) and to identify which aspects are generic and which aspects are technology-specific for optical networks.

This document defines one YANG module: `ietf-network-inventory.yang` (Section 4).

Note: review in future versions of this document the related modules, depending on the augmentation relationship.

The YANG data model defined in this document conforms to the Network Management Datastore Architecture [RFC8342].

1.1. Terminology and Notations

Refer to [RFC7446] and [RFC7581] for the key terms used in this document. The following terms are defined in [RFC7950] and are not redefined here:

- * client
- * server
- * augment
- * data model
- * data node

The following terms are defined in [RFC6241] and are not redefined here:

- * configuration data
- * state data

The terminology for describing YANG data models is found in [RFC7950].

TBD: Recap the concept of chassis/slot/component/board/... in [TMF-MTOSI].

Following terms are used for the representation of the hierarchies in the optical network inventory.

Network Element:

a device installed on one or several shelves and can afford some specific transmission function independently.

Cabinet:

a holder of the device and provides power supply for the device in it.

Chassis:

a holder of the device installation.

Slot:

a holder of the board.

Component:

holders and equipments of the network element, including rack, shelf, slot, sub-slot, board and port.

Board/Card:

a pluggable equipment on the network element and can afford a specific transmission function independently.

Port:

an interface on board

1.2. Tree Diagram

A simplified graphical representation of the data model is used in Section 3 of this document. The meaning of the symbols in these diagrams is defined in [RFC8340].

1.3. Prefix in Data Node Names

In this document, names of data nodes and other data model objects are prefixed using the standard prefix associated with the corresponding YANG imported modules, as shown in the following table.

Prefix	Yang Module	Reference
ianahw	iana-hardware	[RFC8348]
ni	ietf-network-inventory	RFCXXX
yang	ietf-yang-types	[RFC6991]

Table 1: Prefixes and corresponding YANG modules

RFC Editor Note: Please replace XXXX with the RFC number assigned to this document. Please remove this note.

2. YANG Data Model for Optical Network Inventory

2.1. YANG Model Overview

Based on TMF classification in [TMF-MTOSI], inventory objects can be divided into two groups, holder group and equipment group. The holder group contains rack, shelf, slot, sub-slot while the equipment group contains network-element, board and port. With the requirement of GIS and on-demand domain controller selection raised, the equipment room becomes a new inventory object to be managed besides TMF classification.

Logically, the relationship between these inventory objects can be described by Figure 1 below:

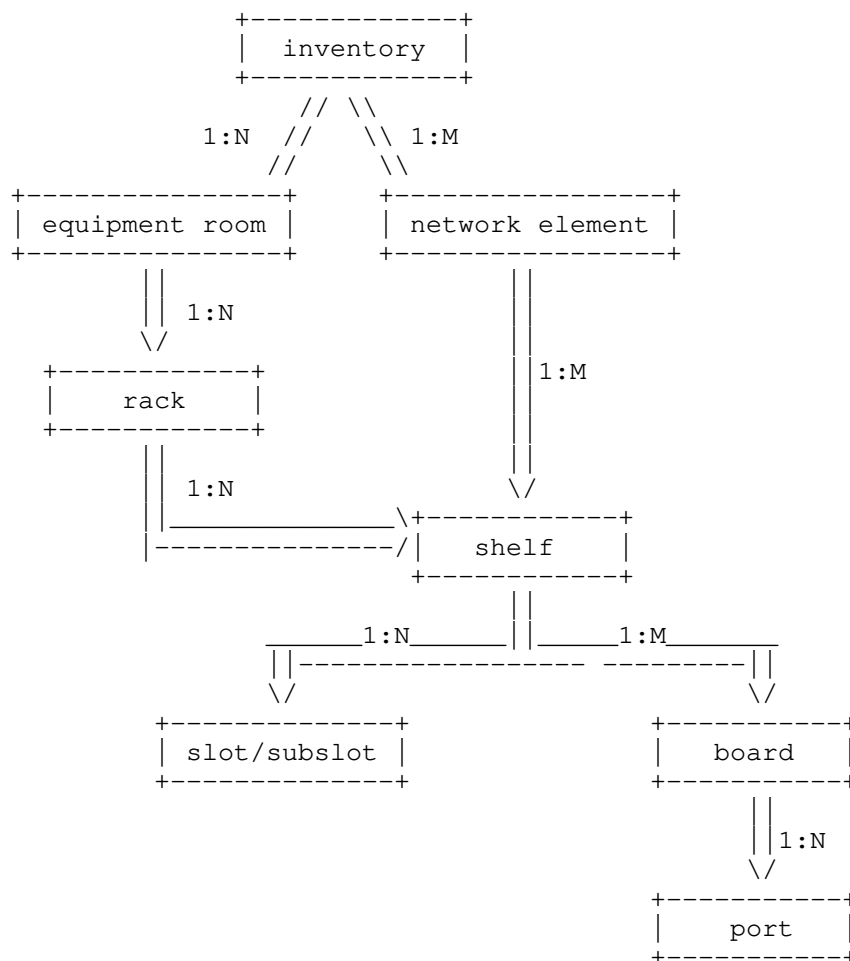


Figure 1: Relationship between inventory objects

In [RFC8348], rack, shelf, slot, sub-slot, board and port are defined as components of network elements with generic attributes.

While [RFC8348] is used to manage the hardware of a single server (e.g., a Network Element), the Network Inventory YANG data model is used to retrieve the network inventory information that a controller discovers from multiple Network Elements under its control.

However, the YANG data model defined in [RFC8348] has been used as a reference for defining the YANG network inventory data model. This approach can simplify the implementation of this network inventory model when the controller uses the YANG data model defined in [RFC8348] to retrieve the hardware configuration from the network elements under its control.

Note: review in future versions of this document which attributes from [RFC8348] are required also for network inventory and whether there are attributes not defined in [RFC8348] which are required for network inventory

Note: review in future versions of this document whether to re-use definitions from [RFC8348] or use schema-mount.

```

+--ro network-inventory
  +--ro equipment-rooms
    +--ro equipment-room* [uuid]
      +--ro uuid          yang:uuid
      .....
    +--ro rack* [uuid]
      +--ro uuid          yang:uuid
      .....
    +--ro shelves* [uuid]
      +--ro uuid          yang:uuid
      .....
    +--ro chassis-ref
      +--ro ne-ref?       leafref
      +--ro component-ref? leafref
  +--ro network-elements
    +--ro network-element* [uuid]
      +--ro uuid          yang:uuid
      .....
    +--ro components
      +--ro component* [uuid]
        +--ro uuid          yang:uuid
        .....

```

The YANG data model for network inventory follows the same approach of [RFC8348] and reports the network inventory as a list of components of different types (e.g., chassis, module, port).

```

+--ro components
  +--ro component* [uuid]
    +--ro uuid          yang:uuid
    +--ro name?         string
    +--ro description?  string
    +--ro class?        identityref
    +--ro parent-rel-pos? int32
    +--ro children* [child-ref]
      | +--ro child-ref  -> ../../uuid
    +--ro parent
      +--ro parent-ref? -> ../../uuid

```

Note: review in future versions of this document whether the component list should be under the network-inventory instead of under the network-element container

However, considering there are some special scenarios, the relationship between the rack and network elements is not 1 to 1 nor 1 to n. The network element cannot be the direct parent node of the rack. So there should be n to m relationship between racks and network elements. And the shelves in the rack should have some reference information to the component.

Note that in [RFC8345], topology and inventory are two subsets of network information. However, considering the complexity of the existing topology models and to have a better extension capability, we define a separate root for the inventory model. We will consider some other ways to do some associations between the topology model and inventory model in the future.

Note: review in future versions of this document whether network inventory should be defined as an augmentation of the network model defined in [RFC8345] instead of under a new network-inventory root.

The proposed YANG data model has been analysed to cover the requirements and use cases for Optical Network Inventory.

Further analysis of requirements and use cases is needed to extend the applicability of this YANG data model to other types of networks (IP and microwave) and to identify which aspects are generic and which aspects are technology-specific for optical networks.

3. Optical Network Inventory Tree Diagram

Figure 2 below shows the tree diagram of the YANG data model defined in module ietf-network-inventory.yang (Section 4).

```

module: ietf-network-inventory
+--ro network-inventory
+--ro equipment-rooms
|   +--ro equipment-room* [uuid]
|   |   +--ro uuid          yang:uuid
|   |   +--ro name?         string
|   |   +--ro location?     string
|   |   +--ro rack* [uuid]
|   |   |   +--ro uuid          yang:uuid
|   |   |   +--ro name?         string
|   |   |   +--ro row-number?  uint32
|   |   |   +--ro rack-number? uint32
|   |   |   +--ro shelves* [uuid]
|   |   |   |   +--ro uuid          yang:uuid
|   |   |   |   +--ro name?         string
|   |   |   |   +--ro shelf-number? uint8
|   |   |   |   +--ro chassis-ref
|   |   |   |   |   +--ro ne-ref?         leafref
|   |   |   |   |   +--ro component-ref?  leafref
+--ro network-elements
+--ro network-element* [uuid]
+--ro uuid          yang:uuid
+--ro name?         string
+--ro components
+--ro component* [uuid]
+--ro uuid          yang:uuid
+--ro name?         string
+--ro description?  string
+--ro class?        identityref
+--ro parent-rel-pos? int32
+--ro children* [child-ref]
|   +--ro child-ref  -> ../../uuid
+--ro parent
+--ro parent-ref?   -> ../../uuid

```

Figure 2: Network inventory tree diagram

4. YANG Model for Optical Network Inventory

```

<CODE BEGINS> file "ietf-network-inventory@2021-11-10.yang"
module ietf-network-inventory {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-network-inventory";
  prefix ni;

  import ietf-yang-types {
    prefix yang;
    reference

```



```
    "RFC6991: Common YANG Data Types.";
}

import iana-hardware {
  prefix ianahw;
  reference
    "RFC 8348: A YANG Data Model for Hardware Management.";
}

organization
  "IETF CCAMP Working Group";
contact
  "WG Web:  <https://datatracker.ietf.org/wg/ccamp/>
  WG List:  <mailto:ccamp@ietf.org>

  Editor:   Chaode Yu
            <yuchaode@huawei.com>

  Editor:   Italo Busi
            <italo.busi@huawei.com>

  Editor:   Aihua Guo
            <aihuaguo.ietf@gmail.com>

  Editor:   Sergio Belotti
            <sergio.belotti@nokia.com>

  Editor:   Jean-Francois Bouquier
            <jeff.bouquier@vodafone.com>

  Editor:   Fabio Peruzzini
            <fabio.peruzzini@telecomitalia.it>";

description
  "This module defines a model for retrieving network inventory.

  The model fully conforms to the Network Management
  Datastore Architecture (NMDA).

  Copyright (c) 2021 IETF Trust and the persons
  identified as authors of the code.  All rights reserved.

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

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

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.";

```
// RFC Ed.: replace XXXX with actual RFC number and remove this
// note.
// RFC Ed.: update the date below with the date of RFC publication
// and remove this note.
```

```
revision 2021-11-10 {
  description
    "Initial revision.";
  reference
    "draft-yg3bp-ccamp-optical-inventory-yang-01: A YANG Data
    Model for Optical Network Inventory.";
}

container network-inventory {
  config false;
  description
    "The top-level container for the network inventory
    information.";
  uses equipment-rooms-grouping;
  uses network-elements-grouping;
}

grouping common-entity-attributes {
  description
    "A set of attributes which are common to all the entities
    (e.g., component, equipment room) defined in this module.";
  leaf uuid {
    type yang:uuid;
    description
      "Uniquely identifies an entity (e.g., component).";
  }
  leaf name {
    type string;
    description
      "A name for an entity (e.g., component), as specified by
      a network manager, that provides a non-volatile 'handle'
      for the entity and that can be modified anytime during the
      entity lifetime.
```

```
        If no configured value exists, the server MAY set the value
        of this node to a locally unique value in the operational
        state.";
    }
}
grouping network-elements-grouping {
    description
        "The attributes of the network elements.";
    container network-elements {
        description
            "The container for the list of network elements.";
        list network-element {
            key uuid;
            description
                "The list of network elements within the network.";
            uses common-entity-attributes;
            uses components-grouping;
        }
    }
}

grouping equipment-rooms-grouping {
    description
        "The attributes of the equipment rooms.";
    container equipment-rooms {
        description
            "The container for the list of equipment rooms.";
        list equipment-room {
            key uuid;
            description
                "The list of equipment rooms within the network.";
            uses common-entity-attributes;
            leaf location {
                type string;
                description
                    "compared with the location information of the other
                    inventory objects, a GIS address is preferred for
                    equipment room";
            }
        }
        list rack {
            key uuid;
            description
                "The list of racks within an equipment room.";
            uses common-entity-attributes;
            leaf row-number {
                type uint32;
                description
                    "Identifies the row within the equipment room where
```

```

        the rack is located.";
    }
    leaf rack-number {
        type uint32;
        description
            "Identifies the physical location of the rack within
             the row.";
    }
    list shelves {
        key uuid;
        description
            "The list of shelves within a rack.";
        uses common-entity-attributes;
        leaf shelf-number {
            type uint8;
            description
                "Identifies the location of the shelf within the
                 rack.";
        }
        container chassis-ref {
            description
                "The reference to the network element component
                 representing this shelf.";
            leaf ne-ref {
                type leafref {
                    path "/ni:network-inventory/ni:network-elements"
                        + "/ni:network-element/ni:uuid";
                }
                description
                    "The reference to the network element containing
                     the component.";
            }
            leaf component-ref {
                type leafref {
                    path "/ni:network-inventory/ni:network-elements"
                        + "/ni:network-element[ni:uuid]"
                        + "=current()/../ne-ref[/ni:components]"
                        + "/ni:component/ni:uuid";
                }
                description
                    "The reference to the component within the network
                     element.";
            }
        }
    }
}
}
}
}

```

```
}

grouping components-grouping {
  description
    "The attributes of the hardware components.";
  container components {
    description
      "The container for the list of components.";
    list component {
      key uuid;
      description
        "The list of components within a network element.";
      uses common-entity-attributes;
      leaf description {
        type string;
        description
          "A textual description of the component.";
        reference
          "RFC 8348: A YANG Data Model for Hardware Management.";
      }
      leaf class {
        type identityref {
          base ianahw:hardware-class;
        }
        description
          "An indication of the general hardware type of the
           component.";
        reference
          "RFC 8348: A YANG Data Model for Hardware Management.";
      }
      leaf parent-rel-pos {
        type int32 {
          range "0 .. 2147483647";
        }
        description
          "An indication of the relative position of this child
           component among all its sibling components. Sibling
           components are defined as components that:

              o share the same value of the 'parent' node and

              o share a common base identity for the 'class' node.";
        reference
          "RFC 8348: A YANG Data Model for Hardware Management.";
      }
      list children {
        key child-ref;
        description
```

```

    "The child components that are physically contained by
    this component.";

leaf child-ref {
  type leafref {
    path "../..//ni:uuid";
  }
  description
    "The reference to the child component.";
}
}
container parent {
  description
    "The parent component that physically contains this
    component.

    If this container is not instantiated, it indicates
    that this component is not contained in any other
    component.

    In the event that a physical component is contained by
    more than one physical component (e.g., double-wide
    modules), this container contains the data of one of
    these components. An implementation MUST use the same
    component every time this container is instantiated.";
  leaf parent-ref {
    type leafref {
      path "../..//ni:uuid";
    }
    description
      "The reference to the parent component.";
  }
}
}
}
}
<CODE ENDS>
```

Figure 3: Network inventory YANG module

5. Manageability Considerations

<Add any manageability considerations>

6. Security Considerations

<Add any security considerations>

7. IANA Considerations

<Add any IANA considerations>

8. References

8.1. Normative References

- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [RFC7446] Lee, Y., Ed., Bernstein, G., Ed., Li, D., and W. Imajuku, "Routing and Wavelength Assignment Information Model for Wavelength Switched Optical Networks", RFC 7446, DOI 10.17487/RFC7446, February 2015, <<https://www.rfc-editor.org/info/rfc7446>>.
- [RFC7581] Bernstein, G., Ed., Lee, Y., Ed., Li, D., Imajuku, W., and J. Han, "Routing and Wavelength Assignment Information Encoding for Wavelength Switched Optical Networks", RFC 7581, DOI 10.17487/RFC7581, June 2015, <<https://www.rfc-editor.org/info/rfc7581>>.
- [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>>.
- [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>>.
- [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>>.

[RFC8348] Bierman, A., Bjorklund, M., Dong, J., and D. Romascanu, "A YANG Data Model for Hardware Management", RFC 8348, DOI 10.17487/RFC8348, March 2018, <<https://www.rfc-editor.org/info/rfc8348>>.

[TMF-MTOSI] TM Forum (TMF), "TMF MTOSI 4.0 Equipment Model", TMF SD2-20_EquipmentModel , 2008, <<https://www.tmforum.org/resources/suite/mtosi-4-0/>>.

8.2. Informative References

[I-D.ietf-teas-actn-poi-applicability] Peruzzini, F., Bouquier, J., Busi, I., King, D., and D. Ceccarelli, "Applicability of Abstraction and Control of Traffic Engineered Networks (ACTN) to Packet Optical Integration (POI)", Work in Progress, Internet-Draft, draft-ietf-teas-actn-poi-applicability-03, 12 July 2021, <<https://www.ietf.org/archive/id/draft-ietf-teas-actn-poi-applicability-03.txt>>.

[ONF_TR-547] Open Networking Foundation (ONF), "TAPI v2.1.3 Reference Implementation Agreement", ONF TR-547 TAPI RIA v1.0 , July 2020, <<https://opennetworking.org/wp-content/uploads/2020/08/TR-547-TAPI-v2.1.3-Reference-Implementation-Agreement-1.pdf>>.

[RFC8345] Clemm, A., Medved, J., Varga, R., Bahadur, N., Ananthakrishnan, H., and X. Liu, "A YANG Data Model for Network Topologies", RFC 8345, DOI 10.17487/RFC8345, March 2018, <<https://www.rfc-editor.org/info/rfc8345>>.

Acknowledgments

The authors of this document would like to thank the authors of [I-D.ietf-teas-actn-poi-applicability] for having identified the gap and requirements to trigger this work.

This document was prepared using kramdown.

Authors' Addresses

Chaode Yu
Huawei Technologies

Email: yuchaode@huawei.com

Italo Busi
Huawei Technologies

Email: italo.busi@huawei.com

Aihua Guo
Futurewei Technologies

Email: aihuaguo.ietf@gmail.com

Sergio Belotti
Nokia

Email: sergio.belotti@nokia.com

Jean-Francois Bouquier
Vodafone

Email: jeff.bouquier@vodafone.com

Fabio Peruzzini
TIM

Email: fabio.peruzzini@telecomitalia.it

Oscar Gonzalez de Dios
Telefonica

Email: oscar.gonzalezdedios@telefonica.com

Victor Lopez
Nokia

Email: victor.lopez@nokia.com