Network Working Group
Internet-Draft
Obsoletes: 6728 (if approved)
Intended status: Standards Track
Expires: September 10, 2020

J. Boyd
ADTRAN
M. Seda
Calix

March 9, 2020

YANG Data Models for the IP Flow Information Export (IPFIX) Protocol, Packet Sampling (PSAMP) Protocol, and Bulk Data Export

draft-boydseda-ipfix-psamp-bulk-data-yang-model-03

Abstract

This document defines a flexible, modular YANG model for packet sampling (PSAMP) and bulk data collection and export via the IPFIX protocol. This new model replaces the model defined in RFC 6728, "Configuration Data Model for the IP Flow Information Export (IPFIX) and Packet Sampling (PSAMP) Protocols". All functionality modeled in RFC 6728 has been carried over to this new model.

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

This document obsoletes RFC 6728 (if approved).

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Bulk data collection is an automated collection of device data that is packaged together and delivered to an IPFIX collector. The IPFIX protocol may be used to transport bulk data such as:

- o Sampled (metered) Packet SAMPling (PSAMP) data: [RFC5476] defines PSAMP operations that a device may implement to sample packets passing through a network element for reporting purposes.

- o Statistics from interfaces and sessions: YANG models define statistics that can be retrieved via protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. These statistics can be streamed using an IPFIX transport to an IPFIX collector that supports analytics tools. An operator may wish to take the bulk data and analyze it for trend analysis purposes or other usages (e.g., collect octet counts every 5 minutes for service level agreement purposes or collect reported device temperature for network health purposes).

IPFIX can also be used to meet the bulk transport requirements of other protocols. For example:

- [BBF.TR-352] ICTP (Inter-Channel Transport Protocol): ICTP uses IPFIX to transport dynamic data (e.g., lease information) across participating NG-PON2 (Next-Generation Passive Optical Network 2) systems.

The YANG data models in this document conform to the Network Management Datastore Architecture (NMDA) defined in [RFC8342].

1.1. Historical Perspective

Below is a historical timeline of IETF IPFIX and YANG RFCs:

- [RFC7011] (2013) defines the IPFIX protocol; it obsoleted RFC 5101 (2008).

- [RFC5476] (2009) defines the PSAMP operations of selection (random selection, deterministic selection or hash-based selection) for capturing or metering packets arriving on a device.

- RFC 6020 (2010) and [RFC7950] (2016) define v1.0 and v1.1 of the YANG data modeling language (respectively), and [RFC8342] (2018) updates RFC 7950 to define NMDA (Network Management Datastore Architecture).

- [RFC6728] (2012) defined a Packet SAMPling (PSAMP) YANG model for devices that use PSAMP for capturing (for metering purposes) a subset of all packets traversing a device.

- [RFC8343] (2018) defines a YANG data model for interfaces; it obsoleted RFC 7223 (2014).

- IETF, IEEE, Broadband Forum etc. (2015 to 2018) have incorporated reporting of statistics into corresponding YANG models (G.fast, PON, etc.).

[RFC6728] defines a single YANG module for the IP Flow Information Export (IPFIX) and Packet Sampling (PSAMP) protocols. The PSAMP collecting process and the IPFIX exporting process are tightly coupled in this module. Moreover, the exporting process requires a device to support SCTP. This coupling and transport requirement makes it difficult for a device, which does not support SCTP, to use the model for collecting and exporting non-PSAMP bulk data.

- o [BBF.TR-352] supports only TCP and TLS as IPFIX transport protocols. The [RFC6728] YANG model does not allow for explicit non-support for SCTP, therefore requiring the need for YANG deviations to announce non-support.

- o The PSAMP meter does not need to be configured if the observation point is already defined by other YANG models. One could attempt to augment PSAMP YANG to reference where the observation point is being configured (but then would have to express feature "non-support" on features unlikely to be needed or required by devices).

Rather than this approach, a new YANG model has been developed where functionality is separated into different modules such that the functions can be independently leveraged.

These are some of the other issues with the current model:

- o The PSAMP YANG model defines the frequency of export in the PSAMP cache. Bulk data needs the export frequency to be controlled by the exporting process.

- o The PSAMP YANG model supports IPFIX mediators. Devices may need to support large IPFIX mediation functions.

- o The PSAMP YANG model contains references which correlate to MIB definitions. For example, interfaces are referenced via ifIndex. For most NETCONF managed devices, interfaces are referenced by name as defined in [RFC8343].

1.2. Relationship to RFC 6728

This RFC adheres to all principles defined in [RFC6728], however, in order to address the issues identified in the previous section, the YANG model has changed as follows:

- o The YANG model is divided into the following three modules:

    * ietf-ipfix: Defines the IPFIX collector and exporter functions.

    * ietf-ipfix-packet-sampling: Defines the PSAMP functions for configuring a device to sample/meter a subset of packets from the network.

    * ietf-ipfix-bulk-data-export: Defines the bulk data IPFIX templates used to export bulk data.

o SCTP data nodes are made optional via the 'sctp' feature for applications not requiring to support SCTP.

o The YANG model adds support for [RFC8343] interface and [RFC8348] hardware component references.

  * The ability to reference via the interface list in ietf-interfaces [RFC8343] is added alongside the ifName and ifIndex.

  * The ability to reference via the hardware component list in ietf-hardware [RFC8348] is added alongside the entPhysicalName and entPhysicalIndex.

o IPFIX transport sessions allow transport session information to be retrieved individually.

  * The transport sessions are modeled such that they can be retrieved individually in addition to retrieving the entire list (which may be quite large for devices such as an NG-PON2 OLT).

o Source and destination address type choice statements are added to improve extensibility of the model.

o This RFC conforms to the [RFC8407] YANG data model guidelines.

Applications that use this RFC are expected to only need to import the applicable YANG modules. For example:

o PSAMP uses the ietf-ipfix and ietf-ipfix-packet-sampling modules.

o Bulk data export uses the ietf-ipfix and ietf-ipfix-bulk-data-export modules.

o Mediators and file readers/writers use only the ietf-ipfix module.

1.3. 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.

The following terms are used in this RFC:

Bulk Data

Bulk data is the collection of configuration and/or state data from a device.

Cache
The Cache is a functional block in a Metering Process that generates IPFIX Flow Records or PSAMP Packet Reports from a Selected Packet Stream, in accordance with its configuration. If Flow Records are generated, the Cache performs tasks like creating new records, updating existing ones, computing Flow statistics, deriving further Flow properties, detecting Flow expiration, passing Flow Records to the Exporting Process, and deleting Flow Records. If Packet Reports are generated, the Cache performs tasks like extracting packet contents and derived packet properties from the Selected Packet Stream, creating new records, and passing them as Packet Reports to the Exporting Process.

Cache Layout
The Cache Layout defines the superset of fields that are included in the Packet Reports or Flow Records maintained by the Cache. The fields are specified by the corresponding Information Elements. In general, the largest possible subset of the specified fields is derived for every Packet Report or Flow Record. More specific rules about which fields must be included are given in Section 4.3.3.

Collector
A device that hosts one or more Collecting Processes is termed a Collector. [RFC7011]

Collecting Process
A Collecting Process received IPFIX messages from one or more Exporting Processes. The Collecting Process might process or store received Flow Records received within these Messages, but such actions are out of scope for this document. [RFC7011]

Composite Selector
A Composite Selector is an ordered composition of Selectors, in which the output Packet Stream issuing from one Selector forms the input Packet Stream to the succeeding Selector. [RFC5476]

Data Record
A Data Record is a record that contains values of the parameters corresponding to a Template Record. [RFC7011]

Exporter
A device that hosts one or more Exporting Process is termed an Exporter. [RFC7011]

Exporting Process
Depending on its deployment as part of an IPFIX Device or PSAMP Device, the Exporting Process sends IPFIX Flow Records or PSAMP Packet Reports to one or more Collecting Processes. The IPFIX Flow Records or PSAMP Packet Reports are generated by one or more Metering Processes.

Filtering
A filter is a Selector that selects a packet deterministically based on the Packet Content, or its treatment, or functions of these occurring in the Selection State. Two examples are:

* Property Match filtering: A packet is selected if the specific field in the packet equals a predefined value.

* Hash-based Selection: A Hash Function is applied to the packet content and the packet is selected if the result falls in a specific range. [RFC5476]

Flow Key
Each of the fields that:

* belong to the packet header (e.g., destination IP address), or

* are a property of the packet itself (e.g., packet length), or

* are derived from Packet Treatment (e.g., Autonomous System (AS) number),

and that are used to define a Flow (i.e., are the properties common to all packets in the Flow) are termed Flow Keys. As an example, the traditional '5-tuple' Flow Key of source and destination IP address, source and destination transport port, and transport protocol, groups together all packets belonging to a single direction of communication on a single socket. [RFC7011]

Flow Record
A Flow Record contains information about a specific Flow that was observed at an Observation Point. A Flow Record contains measured properties of the Flow (e.g., the total number of bytes for all the Flow's packets) and usually contains characteristic properties of the Flow (e.g., source IP address). [RFC7011]

Information Element
An Information Element is a protocol and encoding independent description of an attribute that may appear in an IPFIX record. Information Elements are defined in the [IANA-IPFIX] Registry]. The type associated with an Information Element indicates

constraints on what it may contain and also determines the valid encoding mechanisms for use in IPFIX. [RFC7011]

IPFIX Device
An IPFIX Device hosts at least one Exporting Process. It may host further Exporting Processes as well as arbitrary number of Observation Points and Metering Processes. [RFC7011]

IPFIX File
An IPFIX File is a serialized stream of IPFIX Messages; this stream may be stored in a filesystem or transported using some technique customarily used for files. Any IPFIX Message stream that would be considered valid when transported over one or more of the specified IPFIX transports (Stream Control Transmission Protocol (SCTP), TCP, or UDP) as defined in [RFC7011] is considered an IPFIX File. [RFC5655] extends that definition with recommendations on the construction of IPFIX Files. [RFC5655]

IPFIX File Writer
An IPFIX File Writer is a process that writes IPFIX Files to a filesystem. An IPFIX File Writer operates as an IPFIX Exporting Process as specified in [RFC7011] except as modified by [RFC5655].

IPFIX Mediator
An IPFIX Mediator is an IPFIX Device that provides IPFIX Mediation by receiving a record stream from some data sources, hosting one or more Intermediate Processes to transform that stream, and exporting the transformed record stream into IPFIX Messages via an Exporting Process. In the common case, an IPFIX Mediator receives a record stream from a Collecting Process, but it could also receive a record stream from the data sources not encoded using IPFIX, e.g., in the case of conversion from the Netflow V9 protocol [RFC3954] to IPFIX protocol. [RFC7119]

IPFIX Message
An IPFIX Message is a message that originates at the Exporting Process and carries the IPFIX records of this Exporting Process and whose destination is a Collecting Process. An IPFIX Message is encapsulated at the transport layer. [RFC7011]

Metering Process
The Metering Process is split into two functional blocks:

* Selection Process: A Selection Process takes the Observed Packet Stream as its input and selects a subset of that stream as its output.

* Cache: The Cache is a functional block in a Metering Process that generates IPFIX Flow Records or PSAMP Packet Reports from a Selected Packet Stream, in accordance with its configuration.

The Metering Process generates IPFIX Flow Records or PSAMP Packet Reports, depending on its deployment as part of an IPFIX Device or PSAMP Device. If IPFIX Flow Records are generated, the Metering Process MUST NOT aggregate packets observed at different Observation Domains in the same Flow.

Monitoring Device
A Monitoring Device implements at least one of the functional blocks specified in the context of IPFIX or PSAMP. In particular, the term Monitoring Device encompasses Exporters, Collectors, IPFIX Devices, and PSAMP Devices.

Observation Domain
An Observation Domain is the largest set of Observation Points for which Flow Information can be aggregated by a Metering Process. For example, a router line card may be an Observation Domain if it is composed of several interfaces, each of which is an Observation Point. If the IPFIX Message it generates, the Observation Domain includes it Observation Domain ID, which is unique per Exporting Process. That way, the Collecting Process can identify the specific Observation Domain from the Exporter that sends the IPFIX Messages. Every Observation Point is associated with an Observation Domain. It is RECOMMENDED that Observation Domain IDs also be unique per IPFIX Device. [RFC7011]

Observation Point
An Observation Point is a location in the network where packets can be observed. Examples include a line to which a probe is attached, a shared medium, such as an Ethernet based LAN, a single port of a router, or a set of interfaces (physical or logical) of a router. Note that every Observation Point is associated with an Observation Domain and that one Observation Point may be a superset of several other Observation Points. For example, on Observation Point can be an entire line card. That would be a subset of the individual Observation Points at the line card's interfaces. [RFC7011]

Options Template Record
An Options Template Record is a Template Record that defines the structure and interpretation of fields in a Data Record, including defining how to scope the applicability of the Data Record. [RFC7011]

Options Template/Options Template Set

An Options Template Set is a collection of one or more Options Template Records that have been grouped together in an IPFIX Message. [RFC7011]

Packet Report
Packet Reports comprise a configurable subset of a packet's input to the Selection Process include the packet content, information relating to its treatment (e.g., the output interface) and its associated selection state (e.g., the hash of a packet content). [RFC5476]

Primitive Selector
A Selector is primitive if it is not a Composite Selector. [RFC5476]

PSAMP Device
A PSAMP device is a device hosting at least an Observation Point, a Selection Process and an Exporting Process. Typically corresponding Observation Point(s), Selection Process(es) and Exporting Process(es) are co-located at this device, for example, at a router. [RFC5476]

Reverse Information Element
An Information Element defined as corresponding to a normal (or forward) Information Element, but associated with the reverse direction of a Biflow. [RFC5103]

Sampling
A Selector that is not a filter is called a Sampling operation. This reflects the intuitive notion that if the selection of a packet cannot be determine from its content alone, there must be some type of Sampling taking place. [RFC5476]

Selected Packet Stream
The Selected Packet Stream is the set of all packets selected by a Selection Process.

Selection Process
A Selection Process takes the Observed Packet Stream as its input and selects a subset of that stream as its output. [RFC5476]

Selection Sequence
From all the packets observed at an Observation Point, only a few packets are selected by one or more Selectors. The Selection Sequence is a unique value per Observation Domain describing the Observation Point and the Selector IDs through the packets are selected. [RFC5476]

Selection Sequence Report Interpretation

Each Packet Report contains a selectionSequenceId Information Element that identifies the particular combination of Observation Point and Selector(s) used for its selection. For every selectionSequenceId Information Element in use, the PSAMP Device MUST export a Selection Sequence Report Interpretation using an Options Template. [RFC5476]

Selection Sequence Statistics Report Interpretation

A Selector MAY be used in multiple Selection Sequences. However, each use of a Selector must be independent, so each separate logical instance of a Selector MUST maintain its own individual Selection State and statistics. The Selection Sequence Statistics Report Interpretation MUST include the number of observed packets (Population Size) and the number of packets selected (Sample Size) by each instance of its Primitive Selectors. [RFC5476]

Selection State

A Selection Process may maintain state information for use by the Selection Process. At a given time, the Selection State may depend on packets observed at and before that time, and other variables. Examples include:

* sequence numbers of packets at the input of Selectors

* a timestamp of observation of the packet at the Observation Point

* iterators for pseudorandom number generators

* hash values calculated during selection

* indicators of whether the packet was selected by a given Selector

Selection Processes may change portions of the Selection State as a result of processing a packet. Selection state for a packet is to reflect the state after processing the packet. [RFC5476]

Selector

A Selector defines the action of a Selection Process on a single packet of its input. If selected, the packet becomes an element of the output Packet Stream. The Selector can make use of the following information in determining whether a packet is selected:

* the packet content

* information derived from the packet's treatment at the Observation Point

* any selection state that may be maintained by the Selection Process [RFC5476]

Selector Report Interpretation
An IPFIX Data Record, defined by an Options Template Record, MUST be used to send the configuration details of every Selector in use. The Options Template Record MUST contain:

* selectorId Information Element as the Scope field

* SelectorAlgorithm Information Element [RFC5476]

Template Record
A Template Record defines the structure and interpretation of fields in a Data Record. [RFC7011]

Template/Template Set
A Template Set is a collection of one or more Template Records that have been grouped together in an IPFIX Message. [RFC7011]

Traffic Flow or Flow
A Flow is defined as a set of packets or frames passing an Observation Point in the network during a certain time interval. All packets belonging to a particular Flow have a set of common properties. Each property is defined as the result of applying a function to the values of:

* one or more packet header fields (e.g., destination IP address), transport header fields (e.g., destination port number), or application header fields (e.g., RTP header fields)

* one or more characteristics of the packet itself (e.g., number of MPLS labels, etc.)

* one or more of the fields derived from Packet Treatment (e.g., next-hop IP address, the output interface, etc.)

A packet is defined as belonging to a Flow if it completely satisfies all the defined properties of the Flow. Note that the set of packets represented by a Flow may be empty; that is, a Flow may represent zero or more packets. As sampling is a Packet Treatment, this definition includes packets selected by a sampling mechanism. [RFC7011]

1.4. Tree Diagrams

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

2. Objectives

This document defines a YANG data model for the configuration and state retrieval of basic IPFIX functionality as well as PSAMP and bulk data export applications over IPFIX. The YANG modules in this document conform to the Network Management Datastore Architecture (NMDA) [RFC8342] and [RFC8407] YANG guidelines.

3. Structure of the Configuration Data Model

The reference model described in this RFC describes the following models:

o A PSAMP/IPFIX metered model where a PSAMP/IPFIX device configures a meter that samples packets passing through a device, applies an IPFIX template to those packets, and exports IPFIX templates/data records to an IPFIX collector.

o An IPFIX collector/exporter model where an IPFIX device can:

   * terminate multiple IPFIX sessions to a collection process and then export those IPFIX packets to an external IPFIX collector or

   * read an IPFIX formatted file into the collection process and export that file to a destination location.

o A bulk data model where an IPFIX template is applied to configured reference resource that can export bulk data (e.g., statistics, [BBF.TR-352] ICTP IPFIX data).

Figure 1 illustrates the PSAMP metered UML model for a PSAMP/IPFIX monitoring device. The metering process is contained in the ietf-ipfix-packet-sampling module. The metering process comprises a selection-process and cache that refers to an exporting-process. Further explanations about the relationship between selection-process and cache are given in Section 3.1.1. Section 4.4 describes the exporting-process configuration.

```
   +--------------------------------------------------------------+
   | +--------------------+                    Metering Process   |
   | | Module: ietf-ipfix |                                       |
   | |    -packet-sampling|                                       |
   | |--------------------|<------------------------+             |
   | |                    |<--------+             1 |             |
   | +--------------------+         | +-------------+-----------+ |
   |          ^                     | | list: selection-process | |
   |          |                     | |-------------------------| |
   |        1 |                     | |                         | |
   | +--------+-------------+       | +-----------------+-------+ |
   | | list: observation-pt |       |               ^   |         |
   | |----------------------| selection-process-ref |   |         |
   | |                      +-------|---------------+   |         |
   | +----------------------+       |           0..*    |         |
   |                              1 |                   |         |
   |                        +-------+-------+           |         |
   |                        |  list: cache  |           |         |
   |                        |---------------| 0..1      |         |
   |                        |               |<----------+         |
   |                        +---------------+   cache-ref         |
   +--------------------------------+-----------------------------+
    +--------------------+          |
    | Module: ietf-ipfix |          |
    |--------------------|          |
    |                    |          |
    +--------------------+          |
              ^                     | exporting-process-ref
              |                     |
            1 |                     |
    +---------+---------------+     |
    | list: exporting-process |     |
    |-------------------------|     |
    |                         |<----+
    +-------------------------+ 0..*
```

Figure 1: PSAMP-IPFIX metered model

PSAMP/IPFIX monitoring device implementations usually maintain the separation of various functional blocks, although they do not necessarily implement all of them. The configuration data model enables the setting of commonly available configuration parameters for selection-processes and caches, and supports optional configuration for features like the [RFC2863] IF-MIB and [RFC6933] ENTITY-MIB.

3.1. PSAMP-IPFIX Metered Model

3.1.1. Metering Process Decomposition in Selection Process and Cache

In a monitoring device implementation, the functionality of the metering process is split into the selection process and cache. Figure 2 shows a metering process example. The selection-process takes an observed packet stream as its input and selects a subset of that stream as its output (selected packet stream). The action of the selection-process on a single packet of its input is defined by one selector (called a primitive selector) or an ordered composition of multiple selectors (called a composite selector). The cache generates flow records or packet reports from the selected packet stream, depending on its configuration.

```
              +-------------------------------------+
              | Metering Process                    |
              | +-------------+ Selected            |
    Observed  | | selection-  | Packet    +-------+ |  Stream of
    Packet  -->| process     |---------->| cache |--> Flow Records or
    Stream    | +-------------+ Stream    +-------+ |  Packet Reports
              +-------------------------------------+
```

Figure 2: Selection Process and Cache forming a Metering Process

A metering process must always have a selection-process. It is possible to select all packets in the observed packet stream, and pass them to the cache unfiltered by configuring the selector-method to "select-all".

A metering process can be configured to support multiple selection processes that receive packets from multiple observation points within the same observation domain. In this case, the observed packet streams of the observation points are processed in independent selection sequences. As specified in [RFC5476], a distinct set of selector instances needs to be maintained per selection sequence in order to keep the selection states and statistics separate.

With the configuration data model, it is possible to configure a metering process with multiple selection processes whose output is processed by a single cache. This is illustrated in Figure 3.

```
             +---------------------------------------+
             | Metering Process                      |
             | +------------+ Selected               |
    Observed | | selection- | Packet                 |
    Packet  -->| process    |----------+  +-------+  |
    Stream   | +------------+ Stream   +->|       |  |  Stream of
             |      ...                   | cache |--> Flow Records or
             | +------------+ Selected +->|       |  |  Packet Reports
    Observed | | selection- | Packet   |  +-------+  |
    Packet  -->| process    |----------+             |
    Stream   | +------------+ Stream                 |
             +---------------------------------------+
```

Figure 3: Metering Process with multiple Selection Processes

The observed packet streams at the input of a metering process may originate from observation points belonging to different observation domains. By definition of the observation domain (see [RFC7011]), a cache must not aggregate packets observed at different observation domains in the same flow. Hence, if the cache is configured to generate flow records, it needs to distinguish packets according to their observation domains.

3.1.2. Exporter Configuration

Figure 4 below shows the main classes of the configuration data model that are involved in the configuration of an IPFIX or PSAMP Exporter. The role of the classes can be briefly summarized as follows:

- o The ObservationPoint class specifies an observation-point (e.g., an interface or line card) of the Monitoring Device that captures packets for traffic measurements. An observation-point may be associated with one or more instances of the SelectionProcess class when a device is capable of processing observed packets in parallel.

  * When an observation-point is configured without references to the selection-process, the captured packets are not considered part of the metering process.

- o The SelectionProcess class contains the configuration and state parameters of a selection-process. The selection-process may be composed of a single selector or a sequence of selectors, defining a primitive or composite Selector, respectively. The selection-process selects packets from one or more observed packet streams, each originating from a different observation-point. A selection-process instance may be referred to from one or more observation-point instances.

  * A selection process may pass the selected packet stream to a cache. Therefore, the selection-process class contains a reference to an instance of the cache class.

  * If a selection-process is configured without any reference to a cache, the selected packets are not accounted in any packet report or flow record.

- The Cache class contains configuration and state parameters of a cache. A cache may receive the output of one or more selection processes and maintains corresponding packet reports or flow records. Therefore, an instance of the cache class may be referred to from multiple selection process instances. Configuration parameters of the cache class specify the size of the cache, the cache layout, and expiration parameters if applicable. The cache configuration also determines whether packet reports or flow records are generated.

  * A cache may pass its output to one or more exporting processes. Therefore, the cache class enables references to one or more instances of the exporting process class.

  * If a cache instance does not specify any reference to an exporting process instance, the cache output is dropped.

- The ExportingProcess class contains configuration and state parameters of an exporting-process. It includes various transport-protocol-specific parameters and the export destinations.

  * An instance of the exporting process class may be referred to from multiple instances of the cache class.

```
                        +-------------------+
                        | observation-point |
                        +---------+---------+
                             0..* |
                                  |
                             0..* v
                        +-------------------+
                        | selection-process |
                        +---------+---------+
                             0..* |
                                  |
                             0..1 v
                        +-------------------+
                        | cache             |
                        +---------+---------+
                             0..* |
                                  |
                             0..* v
                        +-------------------+
                        | exporting-process |
                        +-------------------+
```

Figure 4: Class diagram of Exporter configuration

3.2. Collector/Exporter Model

```
     +---------------------+
     | Module: ietf-ipfix  |
     |---------------------|<-------------------+
     |                     |                  1 |
     +---------------------+     +--------------+-----------+
               ^                 | list: collecting-process |
               |                 |--------------------------|
               |                 +--------------+-----------+
               |                                |
             1 |                                |
     +---------+---------------+                |
     | list: exporting-process |                |
     |-------------------------| 0..*           |
     |                         |<---------------+
     +-------------------------+  exporting-process-ref
```

Figure 5: Collector/Exporter Model

3.2.1. Collector/Exporter Decomposition

Figure 5 shows the main classes of the configuration data model that are involved in the configuration of a collector. An instance of the CollectingProcess class specifies the local IP addresses, transport protocols, and port numbers of a collecting-process.

A collecting-process MAY be configured as a File Reader according to [RFC5655].

A CollectingProcess class instance may refer to one or more exporting-process instances configuring exporting processes that re-export the received data. As an example, an exporting process can be configured as a file-writer in order to save the received IPFIX messages in a file.

3.3. Bulk Data Exporter Model

```
    +-------------------------------------+
    | module:ietf-ipfix-bulk-data-export |
    |-------------------------------------|
    +-------------------------------------+
                 ^
                 |
               1 |
    +------------+------------+              +---------------+
    | list:bulk-data-template |              | list:resource |
    |-------------------------|------------->+---------------|
    +------------+------------+         0..* +---------------+
                 |
            0..* | exporting-process-ref
                 v
    +-------------------------+
    | list:exporting-process  |
    |-------------------------|
    +-------------------------+
```

Figure 6: Bulk Data Exporter Model

3.3.1. Bulk Data Exporter Decomposition

Figure 6 shows the main classes of the configuration model that are involved in bulk data export. A device that has a resource instance capable of reporting bulk data through IPFIX does not need an IPFIX meter to be created. Instead a bulk-data template is created and applied to that resource instance.

The ExportingProcess class contains configuration and state parameters of an exporting-process. It includes various transport-protocol-specific parameters and the export destinations. The bulk-data-template may refer to multiple instances of the ExportingProcess class.

4. Configuration and State Parameters

This section specifies the configuration and state parameters of the configuration data model separately for each class.

4.1. Observation Point Class

Figure 7 shows the observation-point attributes of an IPFIX monitoring device. As defined in [RFC7011], an observation point can be any location where packets are observed. A IPFIX monitoring device potentially has more than one such location. An instance of observation-point defines which location is associated with a specific observation point. For this purpose, interfaces (ietf-interfaces module [RFC8343]) and hardware components (ietf-hardware module [RFC8348]) are identified using their names.

- Alternatively, index values of the corresponding entries in the ifTable (IF-MIB module [RFC2863]) or the entPhysicalTable (ENTITY-MIB module [RFC6933]) can be used as identifiers. However, indices should only be used as identifiers if an SNMP agent on the same monitoring device enables access to the corresponding mib tables.

By its definition in [RFC7011], an observation point may be associated with a set of interfaces. Therefore, the configuration data model allows configuring multiple interfaces and hardware components for a single observation point. The observation-point-id (i.e., the value of the information element observationPointId [IANA-IPFIX]) is assigned by the monitoring device.

```
        +--rw observation-point* [name]
           +--rw name                       ietf-ipfix:name-type
           +--rw observation-domain-id    uint32
           +--rw interface-ref*           if:interface-ref
           +--rw if-name*                 if-name-type {if-mib}?
           +--rw if-index*                uint32 {if-mib}?
           +--rw hardware-ref*            hardware-ref
           +--rw ent-physical-name*       string {entity-mib}?
           +--rw ent-physical-index*      uint32 {entity-mib}?
           +--rw direction?               direction
           +--ro observation-point-id?    uint32
           +--rw selection-process*
                   -> /ietf-ipfix:ipfix/psamp/selection-process/name
```

Figure 7: Observation Point Attributes

The configuration parameters of the observation point are:

observation-domain-id
   This parameter defines the identifier of the observation domain that the observation point belongs to. Observation points that are configured with the same observation domain ID belong to the same observation domain. Note that this parameter corresponds to ipfixObservationPointObservationDomainId in the IPFIX MIB module [RFC6615].

interface-ref
   This parameter identifies the interface (via the interface reference [RFC8343]) on the monitoring device that is associated with the given observation point.

if-name
   This parameter identifies the interface (via the ifName in the IF-MIB [RFC2863]) on the monitoring device that is associated with the given observation point. if-name should only be used if an SNMP agent enables access to the ifTable.

if-index
   This parameter identifies the interface (via the ifIndex value in the IF-MIB [RFC2863]) on the monitoring device that is associated with the given observation point. if-index should only be used if an SNMP agent enables access to the ifTable.

hardware-ref
   This parameter identifies a hardware component (via the hardware reference [RFC8348]) on the monitoring device that is associated with the given observation point.

ent-physical-name
This parameter identifies a physical entity (via the entPhysicalName in the ENTITY-MIB module [RFC6933]) on the monitoring device that is associated with the given observation point. ent-physical-name should only be used if an SNMP agent enables access to the entPhysicalTable.

ent-physical-index
This parameter identifies a physical entity (via the entPhysicalIndex in the ENTITY-MIB module [RFC6933]) on the monitoring device that is associated with the given observation point. ent-physical-name should only be used if an SNMP agent enables access to the entPhysicalTable.

direction
This parameter specifies if ingress traffic, egress traffic, or both ingress and egress traffic is captured, using the values "ingress", "egress", and "both", respectively. if not configured, ingress and egress traffic is captured (i.e., the default value is "both"). If not applicable (e.g., in the case of a sniffing interface in promiscuous mode), the value of this parameter is ignored.

selection-process-reference
An observation-point instance may refer to one or more selection-process instances that process the observed packets in parallel.

4.2. Selection Process Class

Figure 8 shows the selection-process attributes. The selection-process class contains the configuration and state parameters of a selection process that selects packets from one or more observed packet streams and generates a selected packet stream as its output. A non-empty ordered list defines a sequence of selectors. The actions defined by the selectors are applied to the stream of incoming packets in the specified order.

If the selection process receives packets from multiple observation points, the observed packet streams need to be processed independently in separate selection sequences. Each selection sequence is identified by a selection sequence id that is unique within the observation domain the observation point belongs to (see [RFC5477]). Selection sequence ids are assigned by the monitoring device.

As state parameters, the selection-process class contains a list of (observation-domain-id, selection-sequence-id) tuples specifying the assigned selection sequence ids and corresponding observation domain ids. With this information, it is possible to associate selection sequence (statistics) report interpretations exported according to the PSAMP protocol specification [RFC5476] with the corresponding selection-process instance.

A selection-process instance may include a reference to a cache class instance to generate packet reports or flow records from the selected packet stream.

```
     +--rw selection-process* [name]
        +--rw name                  ietf-ipfix:name-type
        +--rw selector* [name]
        |  +--rw name
        |  |       ietf-ipfix:name-type
        |  +--rw (method)
        |  |  +--:(select-all)
        |  |  |  +--rw select-all?             empty
        |  |  +--:(samp-count-based)
        |  |  |  ...
        |  |  +--:(samp-time-based)
        |  |  |  ...
        |  |  +--:(samp-rand-out-of-n)
        |  |  |  ...
        |  |  +--:(samp-uni-prob)
        |  |  |  ...
        |  |  +--:(filter-match)
        |  |  |  ...
        |  |  +--:(filter-hash)
        |  |  |  ...
        |  +--ro packets-observed?             yang:counter64
        |  +--ro packets-dropped?              yang:counter64
        |  +--ro selector-discontinuity-time?   yang:date-and-time
        +--rw cache?
        |       -> /ietf-ipfix:ipfix/psamp/cache/name
        +--ro selection-sequence* []
           +--ro observation-domain-id?   uint32
           +--ro selection-sequence-id?   uint64
```

Figure 8: Selection Process Attributes

4.2.1. Selection Process Class Method

Standardized PSAMP sampling and filtering methods are described in [RFC5475]; their configuration parameters are specified in the classes samp-count-based, samp-time-based, samp-rand-out-of-n, samp-uni-prob, filter-match, and filter-hash. In addition, the select-all class, which has no parameters, is used for a selector that selects all packets. The selector class includes exactly one of these sampler and filter classes, depending on the applied method.

```
     +--rw selection-process* [name]
        +--rw name                  ietf-ipfix:name-type
        +--rw selector* [name]
        |  +--rw name
        |  |       ietf-ipfix:name-type
        |  |  ...
        |  +--ro packets-observed?             yang:counter64
        |  +--ro packets-dropped?              yang:counter64
        |  +--ro selector-discontinuity-time?   yang:date-and-time
```

Figure 9: Selector Class Attributes

The selector class, shown in Figure 9 contains the selector statistics packets-observed and packets-dropped as well as selector-discontinuity-time, which correspond to the IPFIX MIB module objects ipfixSelectionProcessStatsPacketsObserved, ipfixSelectionProcessStatsPacketsDropped, and ipfixSelectionProcessStatsDiscontinuityTime, respectively [RFC6615]:

packets-observed
: The total number of packets observed at the input of the selector. If this is the first selector in the selection process, this counter corresponds to the total number of packets in all observed packet streams at the input of the selection process. Otherwise, the counter corresponds to the total number of packets at the output of the preceding selector. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of selector-discontinuity-time.

packets-dropped
: The total number of packets discarded by the selector. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of selector-discontinuity-time.

selector-discontinuity-time
: Timestamp of the most recent occasion at which one or more of the selector counters suffered a discontinuity. In contrast to ipfixSelectionProcessStatsDiscontinuityTime, the time is absolute and not relative to sys-uptime.

Note that packets-observed and packets-dropped are aggregate statistics calculated over all selection sequences of the selection process. This is in contrast to the counter values in the selection sequence statistics report interpretation [RFC5476], which are related to a single selection sequence only.

4.2.1.1. Selection Process Class Method: Sampler Methods

```
     |  |  +--:(samp-count-based)
     |  |  |  +--rw samp-count-based {psamp-samp-count-based}?
     |  |  |     +--rw packet-interval    uint32
     |  |  |     +--rw packet-space       uint32
     |  |  +--:(samp-time-based)
     |  |  |  +--rw samp-time-based {psamp-samp-time-based}?
     |  |  |     +--rw time-interval    uint32
     |  |  |     +--rw time-space       uint32
     |  |  +--:(samp-rand-out-of-n)
     |  |  |  +--rw samp-rand-out-of-n
     |  |  |          {psamp-samp-rand-out-of-n}?
     |  |  |     +--rw size          uint32
     |  |  |     +--rw population    uint32
     |  |  +--:(samp-uni-prob)
     |  |  |  +--rw samp-uni-prob {psamp-samp-uni-prob}?
     |  |  |     +--rw probability    decimal64
```

Figure 10: Sampler Method Attributes

Figure 10 shows the following sampler methods:

samp-count-based (Systematic Count-based Sampling): The following attributes are configurable:

packet-interval
The number of packets that are consecutively sampled between gaps of length packet-space. This parameter corresponds with the Information Element samplingPacketInterval and psampSampCountBasedInterval attribute [RFC5477].

packet-space:
The number of unsampled packets between two sampling intervals. This parameter corresponds to the Information Element samplingPacketSpace and psampSampCountBasedSpace attribute [RFC6727].

Samp-Time-Based (Systematic Time-based Sampling): The following attributes are configurable:

time-interval
The time interval during which all arriving packets are sampled. The unit is microseconds. This parameter corresponds to

corresponds to the Information Element samplingTimeInterval and to psampSampTimeBasedInterval attribute [RFC6727].

time-space
: The gap between two Sampling intervals, in microseconds. This parameter corresponds to Information Element samplingTimeSpace and to psampSampTimeBasedSpace attribute [RFC6727].

Samp-Rand-Out-of-N: The following attributes are configurable:

size
: The number of elements taken from the parent population. This parameter corresponds to Information Element samplingSize and psampSampRandOutOfNSize attribute [RFC6727].

population
: The number of elements in the parent population. These parameters correspond to Information Element samplingPopulation and psampSampRandOutOfNPopulation attribute [RFC6727].

samp-uni-prob: The following attributes are configurable:

probability
: The probability for uniform probabilistic sampling. The probability is expressed as a value between 0 and 1. This parameter corresponds to Information Element samplingProbability and psampSampUniProbProbability attribute [RFC6727].

4.2.2. Selection Process Filter Classes

```
     |  |  +--:(filter-match)
     |  |  |  +--rw filter-match {psamp-filter-match}?
     |  |  |     +--rw (information-element)
     |  |  |     |  +--:(ie-name)
     |  |  |     |  |  +--rw ie-name?
     |  |  |     |  |          ietf-ipfix:ie-name-type
     |  |  |     |  +--:(ie-id)
     |  |  |     |     +--rw ie-id?
     |  |  |     |             ietf-ipfix:ie-id-type
     |  |  |     +--rw ie-enterprise-number?   uint32
     |  |  |     +--rw value                   string
     |  |  +--:(filter-hash)
     |  |     +--rw filter-hash {psamp-filter-hash}?
     |  |        +--rw hash-function?       identityref
     |  |        +--rw initializer-value?   uint64
     |  |        +--rw ip-payload-offset?   uint64
     |  |        +--rw ip-payload-size?     uint64
     |  |        +--rw digest-output?       boolean
     |  |        +--rw selected-range* [name]
     |  |        |  +--rw name    ietf-ipfix:name-type
     |  |        |  +--rw min?    uint64
     |  |        |  +--rw max?    uint64
     |  |        +--ro output-range-min?    uint64
     |  |        +--ro output-range-max?    uint64
```

Figure 11: Filter Method Attributes

Figure 11 shows the following filter methods:

Property-Match Filtering: The following attributes are configurable:

Filtering based on ie-id, ie-name, ie-enterprise-number
   The property to be matched is specified by either ie-id or ie-name, specifying the identifier or name of the Information Element, respectively. If ie-enterprise-number is zero (which is the default), this Information Element is registered in the IANA registry of IPFIX Information Elements [IANA-IPFIX]. A non-zero value of ie-enterprise-number specifies an enterprise specific Information Element [IANA-ENTERPRISE-NUMBERS].

value
   The matching value.

For hash-based filtering, the configuration and state attributes are:

hash-function
   The following values are defined:

* BOB: BOB Hash Function as specified in [RFC5475], Appendix A.2

* IPSX: IP Shift-XOR (IPSX) Hash Function as specified in [RFC5475], Appendix A.1

* CRC: CRC-32 function as specified in [RFC1141] Default value is "BOB". This parameter corresponds to the PSAMP MIB object psampFiltHashFunction [RFC6727].

initializer-value
This parameter corresponds to the Information Element hashInitialiserValue [RFC5477], as well as to the PSAMP MIB object psampFiltHashInitializerValue [RFC6727]. If not configured by the user, the Monitoring Device arbitrarily chooses an initializer value.

ip-payload-offset
Configures the offset of the payload section used as input to the hash function. Default value is 0 (minimum configurable values according to [RFC5476], Section 6.5.2.6.). This parameter corresponds to the Information Element hashIPPayloadOffset [RFC5477] as well as to the PSAMP MIB object psampFiltHashIpPayloadOffset [RFC6727].

ip-payload-size
Configures the size of the payload section used as input to the hash function. Default value is 8 (minimum configurable values according to [RFC5476], Section 6.5.2.6.). This parameter corresponds to the Information Element hashIPPayloadSize [RFC5477], as well as to the PSAMP MIB object psampFiltHashIpPayloadSize [RFC6727].

digest-output
Enables or disables the inclusion of the packet digest in the resulting PSAMP Packet Report. This requires that the Cache Layout of the Cache generating the Packet Reports includes a digest-hash-value field. This parameter corresponds to the Information Element hashDigestOutput [RFC5477].

output-range-min
Defines the beginning of the hash's function potential output range. This parameter correspond to the Information Element hashOutputRangeMin [RFC5477], as well as to the PSAMP MIB object psampFiltHashOutputRangeMin [RFC6727].

output-range-max
Defines the end of the hash function's potential output range. This parameter correspond to the Information Element

hashOutputRangeMax [RFC5477], as well as to the PSAMP MIB object psampFiltHashOutputRangeMax [RFC6727].

One or more ranges of matching hash values are defined by the min and max parameters of the selected-range subclass. These parameters correspond to the Information Elements hashSelectedRangeMin and hashSelectedRangeMax [RFC5477], as well as to the PSAMP MIB objects psampFiltHashSelectedRangeMin and psampFiltHashSelectedRangeMax [RFC6727].

4.3. Cache Class

Figure 12 shows the cache class that contains the configuration and state parameters of a cache. Most of these parameters are specific to the type of the cache and therefore contained in the subclasses immediate-cache, timeout-cache, natural-cache, and permanent-cache, which are presented below in Section 4.3.1 and Section 4.3.2.

```
+--rw cache* [name]
   +--rw name                         ietf-ipfix:name-type
   +--rw enabled                      boolean
   +--ro metering-process-id?         uint32
   +--ro data-records?                yang:counter64
   +--ro cache-discontinuity-time?    yang:date-and-time
   +--rw (cache-type)
   |  +--:(immediate-cache)
   |  |     ...
   |  +--:(timeout-cache)
   |  |     ...
   |  +--:(natural-cache)
   |  |     ...
   |  +--:(permanent-cache)
   |  |     ...
   +--rw exporting-process*
           -> /ietf-ipfix:ipfix/exporting-process/name
           {ietf-ipfix:exporter}?
```

Figure 12: Cache Attributes

The following configuration and state parameters are common to all caches and therefore included in the cache class itself:

enabled
   Enables the cache so that specified data may be exported. The default is "enabled".

metering-process-id

The identifier of the metering process that cache belongs to. This parameter corresponds to the information element meteringProcessId [IANA-IPFIX]. Its occurrence helps to associate metering process (reliability) statistics exported according to the IPFIX protocol specification [RFC7011] with the corresponding MeteringProcess class identifier.

data-records
: The number of data records generated by this cache.

discontinuities
: The value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of cache-discontinuity-time. Note that this parameter corresponds to ipfixMeteringProcessDataRecords in the IPFIX MIB module [RFC6615].

cache-discontinuity-time
: The timestamp of the most recent occasion at which datarecords suffered a discontinuity. The time is absolute and not relative to sysUpTime. Note that this parameter functionally corresponds to ipfixMeteringProcessDiscontinuityTime in the IPFIX MIB module [RFC6615].

A cache object may refer to one or more exporting-process instances.

4.3.1. Immediate Cache Type Class

The immediate-cache type class depicted in Figure 13 is used to configure a cache that generates a PSAMP Packet Report for each packet at its input. The fields contained in the generated data records are defined in an object of the cache-layout, which is defined below in Section 4.3.3.

```
           +--rw (cache-type)
           |  +--:(immediate-cache)
           |  |  +--rw immediate-cache {immediate-cache}?
           |  |     +--rw cache-layout
           |  |        +--rw cache-field* [name]
           |  |           +--rw name
           |  |           |       ietf-ipfix:name-type
           |  |           +--rw (information-element)
           |  |           |  +--:(ie-name)
           |  |           |  |  +--rw ie-name?
           |  |           |  |          ietf-ipfix:ie-name-type
           |  |           |  +--:(ie-id)
           |  |           |     +--rw ie-id?
           |  |           |             ietf-ipfix:ie-id-type
           |  |           +--rw ie-length?                uint16
           |  |           +--rw ie-enterprise-number?     uint32
           |  |           +--rw is-flow-key?              empty
```

Figure 13: Immediate Cache Attributes

4.3.2. Timeout Cache, Natural Cache, and Permanent Cache Type Class

Figure 14 shows the timeout-cache, natural-cache, and permanent-cache type classes. These classes are used to configure a cache that aggregates the packets at its input and generates IPFIX flow records.

```
           +--rw (cache-type)
           |  +--:(timeout-cache)
           |  |  +--rw timeout-cache {timeout-cache}?
           |  |     +--rw max-flows?              uint32
           |  |     +--rw active-timeout?         uint32
           |  |     +--rw idle-timeout?           uint32
           |  |     +--rw export-interval?        uint32
           |  |     +--rw cache-layout
           |  |     |  ...
           |  |     +--ro active-flows?           yang:gauge32
           |  |     +--ro unused-cache-entries?   yang:gauge32
           |  +--:(natural-cache)
           |  |  +--rw natural-cache {natural-cache}?
           |  |     { same as timeout-cache }
           |  +--:(permanent-cache)
           |     +--rw permanent-cache {permanent-cache}?
           |        { same as timeout-cache }
```

Figure 14: Timeout, Natural and Permanent Cache Attributes

The three classes differ in when flows expire:

timeout-cache
Flows expire after active or idle timeout.

natural-cache
Flows expire after active or idle timeout, or on natural termination (e.g., TCP FIN or TCP RST) of the flow.

permanent-cache
Flows never expire, but are periodically exported with the interval set by export-interval.

The following configuration and state parameters are common to the three classes:

max-flows
This parameter configures the maximum number of entries in the cache, which is the maximum number of flows that can be measured simultaneously. If this parameter is configured, the monitoring device must ensure that sufficient resources are available to store the configured maximum number of flows. If the maximum number of cache entries is in use, no additional flows can be measured. However, traffic that pertains to existing flows can continue to be measured.

active-flows
This state parameter indicates the number of flows currently active in this cache (i.e., the number of cache entries currently in use). Note that this parameter corresponds to ipfixmeteringprocesscacheactiveflows in the IPFIX MIB module [RFC6615].

unused-cache-entries
The number of unused cache entries. Note that the sum of active-flows and unused-cache-entries equals max-flows if max-flows is configured. Note that this parameter corresponds to ipfixMeteringProcessCacheUnusedCacheEntries in the IPFIX MIB module [RFC6615].

The following timeout parameters are only available in the timeout-cache and the natural-cache cache-types:

active-timeout
This parameter configures the time in seconds after which a flow is expired even though packets matching this flow are still received by the cache. The parameter value zero indicates infinity, meaning that there is no active timeout. If not configured by the user, the monitoring device sets this parameter. Note that this parameter corresponds to

ipfixMeteringProcessCacheActiveTimeout in the IPFIX MIB module [RFC6615].

idle-timeout
This parameter configures the time in seconds after which a flow is expired if no more packets matching this flow are received by the cache. The parameter value zero indicates infinity, meaning that there is no idle timeout. If not configured by the user, the monitoring device sets this parameter. Note that this parameter corresponds to ipfixMeteringProcessCacheIdleTimeout in the IPFIX MIB module [RFC6615].

The following interval parameter is only available in the permanent-cache class:

export-interval
This parameter configures the interval (in seconds) for periodical export of flow records. If not configured by the user, the monitoring device sets this parameter.

Every generated flow record must be associated with a single observation domain. Hence, although a cache may be configured to process packets observed at multiple observation domains, the cache must not aggregate packets observed at different observation domains in the same flow.

An object of the cache class contains an object of the cache-layout class that defines which fields are included in the flow records.

4.3.3. Cache Layout Class

A cache generates and maintains packet reports or flow records containing information that has been extracted from the incoming stream of packets. Using the cache-field class, the cache-layout class specifies the superset of fields that are included in the packet reports or flow records (see Figure 15).

If packet reports are generated (i.e., if immediate-cache class is used to configure the cache), every field specified by the cache-layout must be included in the resulting packet report unless the corresponding information element is not applicable or cannot be derived from the content or treatment of the incoming packet. Any other field specified by the cache layout may only be included in the packet report if it is obvious from the field value itself or from the values of other fields in same packet report that the field value was not determined from the packet.

For example, if a field is configured to contain the TCP source port (information element tcpSourcePort [IANA-IPFIX]), the field must be included in all packet reports that are related to TCP packets. Although the field value cannot be determined for non-TCP packets, the field may be included in the packet reports if another field contains the transport protocol identifier (information element protocolIdentifier [IANA-IPFIX]).

If flow records are generated (i.e., if timeout-cache, natural-cache, or permanent-cache class is used to configure the cache), the cache layout differentiates between flow key fields and non-key fields. Every flow key field specified by the cache layout must be included as flow key in the resulting flow record unless the corresponding information element is not applicable or cannot be derived from the content or treatment of the incoming packet. Any other flow key field specified by the cache layout may only be included in the flow record if it is obvious from the field value itself or from the values of other flow key fields in the same flow record that the field value was not determined from the packet. Two packets are accounted by the same flow record if none of their flow key fields differ. If a flow key field can be determined for one packet but not for the other, the two packets are accounted in different flow records.

Every non-key field specified by the cache layout must be included in the resulting flow record unless the corresponding information element is not applicable or cannot be derived for the given flow. Any other non-key field specified by the cache layout may only be included in the flow record if it is obvious from the field value itself or from the values of other fields in same flow record that the field value was not determined from the packet. Packets which are accounted by the same flow record may differ in their non-key fields, or one or more of the non-key fields can be undetermined for all or some of the packets.

For example, if a non-key field specifies an information element whose value is determined by the first packet observed within a flow (which is the default rule according to [RFC7012] unless specified differently in the description of the information element), this field must be included in the resulting flow record if it can be determined from the first packet of the flow.

```
     |  |     +--rw cache-layout
     |  |     |  +--rw cache-field* [name]
     |  |     |     +--rw name
     |  |     |     |       ietf-ipfix:name-type
     |  |     |     +--rw (information-element)
     |  |     |     |  +--:(ie-name)
     |  |     |     |  |  +--rw ie-name?
     |  |     |     |  |          ietf-ipfix:ie-name-type
     |  |     |     |  +--:(ie-id)
     |  |     |     |     +--rw ie-id?
     |  |     |     |             ietf-ipfix:ie-id-type
     |  |     |     +--rw ie-length?              uint16
     |  |     |     +--rw ie-enterprise-number?   uint32
     |  |     |     +--rw is-flow-key?            empty
```

Figure 15: Cache Field Attributes

The cache-layout class does not have any parameters. The configuration parameters of the cache-field class (see Figure 15) are as follows:

ie-name
Specifies the information element name to be used. Either ie-id or ie-name must be specified.

ie-id
Specifies the information element identifier to be used. Either ie-id or ie-name must be specified.

ie-length
This parameter specifies the length of the field in octets. A value of 65535 means that the field is encoded as a variable-length information element. For information elements of integer and float type, the field length may be set to a smaller value than the standard length of the abstract data type if the rules of reduced size encoding are fulfilled (see [RFC7011], section 6.2). If not configured by the user, the field length is set by the monitoring device.

ie-enterprise-number
Specifies the enterprise ID of the ie-id or ie-name. If the ie-enterprise-number is zero (which is the default), this information element is registered in the IANA registry of IPFIX information elements [IANA-IPFIX]. A non-zero value of ie-enterprise-number specifies an enterprise-specific information element [IANA-ENTERPRISE-NUMBERS]. If the enterprise number is set to 29305, this field contains a reverse information element. In this

case, the cache must generate data records in accordance to [RFC5103].

is-flow-key
If present, this field is a flow key. If the field contains a reverse information element, it must not be configured as flow key. This parameter is not available if the cache is configured using the immediate-cache class since there is no distinction between flow key fields and non-key fields in packet reports.

Note that the use of information elements can be restricted to certain cache types as well as to flow key or non-key fields. Such restrictions may result from information element definitions or from device-specific constraints. According to Section 5, the monitoring device must notify the user if a cache field cannot be configured with the given information element.

4.4. Exporting Process Class

The ExportingProcess class in Figure 16) specifies destinations to which the incoming packet reports and flow records are exported using objects of the destination class. The destination class includes a choice of type of exporter (sctp-exporter, udp-exporter, tcp-exporter, or file-writer) which contains further configuration parameters. Those exporter type classes are described in Section 4.4.1, Section 4.4.2, Section 4.4.3, and Section 4.4.4.

The ExportingProcess class contains the identifier of the exporting process (exporting-process-id). This parameter corresponds to the information element exportingProcessId [IANA-IPFIX]. Its occurrence helps to associate exporting process reliability statistics exported according to the IPFIX protocol specification [RFC7011] with the corresponding object of the ExportingProcess class.

The order in which destination instances appear has a specific meaning only if the export-mode parameter is set to "fallback".

```
        +--rw exporting-process* [name] {exporter}?
           +--rw name                     name-type
           +--rw enabled?                 boolean
           +--rw export-mode?             identityref
           +--rw destination* [name]
           |  +--rw name                     name-type
           |  +--rw (destination-parameters)
           |     +--:(tcp-exporter)
           |     |     ...
           |     +--:(udp-exporter)
           |     |     ...
           |     +--:(sctp-exporter)
           |     |     ...
           |     +--:(file-writer)
           |           ...
           +--rw options* [name]
           |  +--rw name               name-type
           |  +--rw options-type       identityref
           |  +--rw options-timeout?   uint32
           +--ro exporting-process-id?   uint32
```

Figure 16: Exporting Process Class

The Exporting Process parameters are defined as follows:

enabled
Enables the exporting process to begin exporting data. The default is "enabled".

export-mode
Determines to which configured destination(s) the incoming data records are exported. The following parameter values are specified by the configuration data model:

* parallel: every data record is exported to all configured destinations in parallel

* load-balancing: every data record is exported to exactly one configured destination according to a device-specific load-balancing policy

* fallback: every data record is exported to exactly one configured destination according to the fallback policy described below

If export-mode is set to "fallback", the first destination instance defines the primary destination, the second destination instance defines the secondary destination, and so on. If the exporting process fails to export data records to the primary destination, it tries to export them to the secondary one. If the secondary destination fails as well, it continues with the tertiary, etc. "parallel" is the default value if exportmode is not configured.

Note that the export-mode parameter is related to the ipfixExportMemberType object in [RFC6615]. If export-mode is "parallel", the ipfixExportMemberType values of the corresponding entries in IpfixExportTable are set to parallel(3). If export-mode is "load-balancing", the ipfixExportMemberType values of the corresponding entries in IpfixExportTable are set to loadBalancing(4). If exportmode is "fallback", the ipfixExportMemberType value that refers to the primary destination is set to primary(1); the ipfixExportMemberType values that refer to the remaining destinations need to be set to secondary(2). The IPFIX mib module does not define any value for tertiary destination, etc.

The reporting of information with options templates is defined with objects of the Options class.

The exporting process may modify the packet reports and flow records to enable a more efficient transmission or storage under the condition that no information is changed or suppressed. For example, the exporting process may shorten the length of a field according to the rules of reduced size encoding [RFC7011]. The exporting process may also export certain fields in a separate data record as described in [RFC5476].

4.4.1. SCTP Exporter Class

The SctpExporter class shown in Figure 17 contains the configuration parameters of an SCTP export destination.

```
        +--:(sctp-exporter)
           +--rw sctp-exporter {sctp-transport}?
              +--rw ipfix-version?              uint16
              +--rw destination-port?
              |       inet:port-number
              +--rw send-buffer-size?           uint32
              +--rw rate-limit?                 uint32
              +--rw transport-layer-security!
              |     ...
              +--rw source
              |  +--rw (source-method)?
              |     +--:(source-address)
              |     |  +--rw source-address?   inet:host
              |     +--:(interface-ref)
              |     |  +--rw interface-ref?    if:interface-ref
              |     +--:(if-index) {if-mib}?
              |     |  +--rw if-index?         uint32
              |     +--:(if-name) {if-mib}?
              |        +--rw if-name?          string
              +--rw destination
              |  +--rw (destination-method)
              |     +--:(destination-address)
              |        +--rw destination-address?   inet:host
              +--rw timed-reliability?          uint32
              +--ro transport-session
                    ...
```

Figure 17: SCTP Exporter Class

The configuration parameters are:

ipfix-version
   Version number of the IPFIX protocol used. If omitted, the default value is 10 (=0x000a) as specified in [RFC7011].

source-address
   List of source IP addresses used by the exporting process. If configured, the specified addresses are eligible local IP addresses of the multihomed SCTP endpoint. If not configured, all locally assigned IP addresses are eligible local IP addresses.

destination-address
   One or more IP addresses of the collecting process to which IPFIX Messages are sent. The user must ensure that all configured IP addresses belong to the same collecting process. The exporting process tries to establish an SCTP association to any of the configured destination IP addresses.

destination-port
Destination port number to be used. If not configured, standard port 4739 (IPFIX without TLS and DTLS) or 4740 (IPFIX over TLS or DTLS) is used.

if-index
The index of the interface used by the exporting process to export IPFIX Messages to the given destination MAY be specified according to corresponding objects in the IF-MIB [RFC2863]. If omitted, the Exporting Process selects the outgoing interface based on local routing decision and accepts return traffic, such as transport-layer acknowledgments, on all available interfaces.

if-name
The name of the interface used by the exporting process to export IPFIX Messages to the given destination MAY be specified according to corresponding objects in the IF-MIB [RFC2863]. If omitted, the Exporting Process selects the outgoing interface based on local routing decision and accepts return traffic, such as transport-layer acknowledgments, on all available interfaces.

send-buffersize
Size of the socket send buffer in bytes. If not configured by the user, the buffer size is set by the monitoring device.

rate-limit
Maximum number of bytes per second the exporting process may export to the given destination as required by [RFC5476]. The number of bytes is calculated from the lengths of the IPFIX Messages exported. If this parameter is not configured, no rate limiting is performed for this destination.

timed-reliability
Lifetime in milliseconds until an IPFIX message containing data sets only is "abandoned" due to the timed reliability mechanism of the partial reliability extension of SCTP (pr-SCTP) [RFC3758]. if this parameter is set to zero, reliable SCTP transport must be used for all data records. Regardless of the value of this parameter, the exporting process may use reliable SCTP transport for data sets associated with certain options templates, such as the data record reliability options template specified in [RFC6526].

Using the TransportLayerSecurity class described in Section 4.6, Datagram Transport Layer Security (DTLS) is enabled and configured for this export destination.

The TransportSession class is discussed in Section 4.7.

4.4.2. UDP Exporter Class

The UdpExporter class shown in Figure 18 contains the configuration parameters of a UDP export destination. The parameters ipfix-version, destination-port, if-name, if-index, send-buffer-size, and rate-limit have the same meaning as in the SctpExporter class (see Section 4.4.1).

```
+--:(udp-exporter)
   +--rw udp-exporter {udp-transport}?
      +--rw ipfix-version?                       uint16
      +--rw destination-port?
      |       inet:port-number
      +--rw send-buffer-size?                    uint32
      +--rw rate-limit?                          uint32
      +--rw transport-layer-security!
      |     ...
      +--rw source
      |  +--rw (source-method)?
      |     +--:(source-address)
      |     |  +--rw source-address?   inet:host
      |     +--:(interface-ref)
      |     |  +--rw interface-ref?    if:interface-ref
      |     +--:(if-index) {if-mib}?
      |     |  +--rw if-index?         uint32
      |     +--:(if-name) {if-mib}?
      |        +--rw if-name?          string
      +--rw destination
      |  +--rw (destination-method)
      |     +--:(destination-address)
      |        +--rw destination-address?   inet:host
      +--rw maximum-packet-size?                 uint16
      +--rw template-refresh-timeout?            uint32
      +--rw options-template-refresh-timeout?   uint32
      +--rw template-refresh-packet?             uint32
      +--rw options-template-refresh-packet?    uint32
      +--ro transport-session
         ....
```

Figure 18: UDP Exporter Class

The remaining configuration parameters are:

source-address
   This parameter specifies the source IP address used by the exporting process. If this parameter is omitted, the IP address assigned to the outgoing interface is used as the source IP address.

destination-address
Destination IP address to which IPFIX messages are sent (i.e., the IP address of the collecting process).

max-packet-size
This parameter specifies the maximum size of IP packets sent to the collector. If set to zero, the exporting device must derive the maximum packet size from path mtu discovery mechanisms. If not configured by the user, this parameter is set by the monitoring device.

template-refresh-timeout
This parameter specifies when templates are refreshed by the exporting process. This timeout is specified in seconds between re-sending of templates. If omitted, the default value of 600 seconds (10 minutes) is used [RFC7011]. This parameter corresponds to ipfixTransportSessionTemplateRefreshTimeout in the IPFIX MIB module [RFC6615].

options-template-refresh-timeout
This parameter specifies when options templates are refreshed by the exporting process. This timeout is specified in seconds between re-sending of options templates. If omitted, the default value of 600 seconds (10 minutes) is used [RFC7011]. This parameter corresponds to ipfixTransportSessionOptionsTemplateRefreshTimeout in the IPFIX MIB module [RFC6615].

template-refresh-packet
This parameter specifies the number of IPFIX messages after which templates are re-sent. If omitted, the templates are only resent after timeout. This parameter corresponds to ipfixTransportSessionTemplateRefreshTimeout in the IPFIX MIB module [RFC6615].

options-template-refresh-packet
This parameter specifies the number of IPFIX messages after which options templates are re-sent. If omitted, the options templates are only resent after timeout. This parameter corresponds to ipfixTransportSessionOptionsTemplateRefreshTimeout in the IPFIX MIB module [RFC6615].

Note that the values configured for template-refresh-timeout and options-template-refresh-timeout must be adapted to the template-lifetime and options-template-lifetime parameter settings at the receiving collecting process (see Section 4.5.2).

Using the TransportLayerSecurity class described in Section 4.6, DTLS is enabled and configured for this export destination. The TransportSession class is specified in Section 4.7.

4.4.3. TCP Exporter Class

The TcpExporter class shown in Figure 19 contains the configuration parameters of a TCP export destination. The parameters have the same meaning as in the UdpExporter class (see Section 4.4.2).

Using the TransportLayerSecurity class described in Section 4.6, Transport Layer Security (TLS) is enabled and configured for this export destination.

The TransportSession class is specified in Section 4.7.

```
+--:(tcp-exporter)
   +--rw tcp-exporter {tcp-transport}?
      +--rw ipfix-version?               uint16
      +--rw destination-port?
      |       inet:port-number
      +--rw send-buffer-size?            uint32
      +--rw rate-limit?                  uint32
      +--rw transport-layer-security!
      |     ...
      +--rw source
      |  +--rw (source-method)?
      |     +--:(source-address)
      |     |  +--rw source-address?   inet:host
      |     +--:(interface-ref)
      |     |  +--rw interface-ref?    if:interface-ref
      |     +--:(if-index) {if-mib}?
      |     |  +--rw if-index?         uint32
      |     +--:(if-name) {if-mib}?
      |        +--rw if-name?          string
      +--rw destination
      |  +--rw (destination-method)
      |     +--:(destination-address)
      |        +--rw destination-address?   inet:host
      +--ro transport-session
```

Figure 19: TCP Exporter Class

4.4.4. File Writer Class

If file-writer instance is included in an object of the destination class, IPFIX messages are written into a file as specified in [RFC5655].

```
        +--:(file-writer)
           +--rw file-writer {file-writer}?
              +--rw ipfix-version?       uint16
              +--rw file                 inet:uri
              +--ro file-writer-state
                 +--ro bytes?
                 |       yang:counter64
                 +--ro messages?
                 |       yang:counter64
                 +--ro discarded-messages?
                 |       yang:counter64
                 +--ro records?
                 |       yang:counter64
                 +--ro templates?
                 |       yang:counter32
                 +--ro options-templates?
                 |       yang:counter32
                 +--ro file-writer-discontinuity-time?
                 |       yang:date-and-time
                 +--ro template* []
                    +--ro observation-domain-id?         uint32
                    +--ro template-id?                   uint16
                    +--ro set-id?                        uint16
                    +--ro access-time?
                    |       yang:date-and-time
                    +--ro template-data-records?
                    |       yang:counter64
                    +--ro template-discontinuity-time?
                    |       yang:date-and-time
                    +--ro field* []
                       +--ro ie-id?                  ie-id-type
                       +--ro ie-length?              uint16
                       +--ro ie-enterprise-number?   uint32
                       +--ro is-flow-key?            empty
                       +--ro is-scope?               empty
```

Figure 20: File Writer Class

The FileWriter class contains the following configuration parameters:

ipfix-version
   Version number of the IPFIX protocol used. If omitted, the default value is 10 (=0x000a) as specified in [RFC7011].

file
   File name and location specified as URI.

The state parameters of the FileWriter class are:

bytes, messages, records, templates, options-templates
The number of bytes, IPFIX messages, data records, template records, and options template records written by the file writer. Discontinuities in the values of these counters can occur at re-initialization of the management system, and at other times as indicated by the value of file-writer-discontinuity-time.

discarded-messages
The number of IPFIX messages that could not be written by the file writer due to internal buffer overflows, limited storage capacity, etc. Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of file-writer-discontinuity-time.

file-writer-discontinuity-time
Timestamp of the most recent occasion at which one or more file writer counters suffered a discontinuity. The time is absolute and not relative to sysUpTime.

Each FileWriter class instance includes statistics about the templates written to the file. The Template class is specified in Section 4.8.

4.4.5. Options Class

The Options class in Figure 21 defines the type of specific information to be reported, such as statistics, flow keys, sampling and filtering parameters, etc. [RFC7011] and [RFC5476] specify several types of reporting information that may be exported.

```
+--rw options* [name]
   +--rw name               name-type
   +--rw options-type       identityref
   +--rw options-timeout?   uint32
```

Figure 21: Options Class

The following parameter values are specified by the configuration data model:

metering-statistics
Export of metering process statistics using the metering process statistics options template [RFC7011].

metering-reliability
Export of metering process reliability statistics using the metering process reliability statistics options template [RFC7011].

exporting-reliability
Export of exporting process reliability statistics using the exporting process reliability statistics options template [RFC7011].

flow-keys
Export of the flow key specification using the flow keys options template [RFC7011].

selection-sequence
Export of selection sequence report interpretation and selector report interpretation [RFC5476].

selection-statistics
Export of selection sequence statistics report interpretation [RFC5476].

accuracy
Export of accuracy report interpretation [RFC5476].

reducing-redundancy
Enables the utilization of options templates to reduce redundancy in the exported data records according to [RFC5473]. The exporting process decides when to apply these options templates.

extended-type-information
Export of extended type information for enterprise-specific information elements used in the exported templates [RFC5610].

The exporting process must choose a template definition according to the options type and available options data. The options-timeout parameter specifies the reporting interval (in milliseconds) for periodic export of the option data. A parameter value of zero means that the export of the option data is not triggered periodically, but whenever the available option data has changed. this is the typical setting for options types flow-keys, selection-sequence, accuracy, and reducing-redundancy. If options-timeout is not configured by the user, it is set by the monitoring device.

4.5. Collecting Process Class

Figure 22 shows the CollectingProcess class that contains the configuration and state parameters of a collecting process. The sctp-collector, udp-collector, and TcpCollector classes specify how IPFIX messages are received from remote exporters. The collecting process can also be configured as a file reader using the FileReader class. These classes are described in Section 4.5.1, Section 4.5.2, Section 4.5.3, and Section 4.5.4.

A collecting-process instance may refer to one or more exporting-process instances configuring exporting processes that export the received data without modifications to a file or to another remote collector.

```
     +--rw collecting-process* [name] {collector}?
        +--rw name                 name-type
        +--rw tcp-collector* [name] {tcp-transport}?
           ...
        +--rw udp-collector* [name] {udp-transport}?
           ...
        +--rw sctp-collector* [name] {sctp-transport}?
           ...
        +--rw file-reader* [name] {file-reader}?
           ...
        +--rw exporting-process*   -> /ipfix/exporting-process/name
                {exporter}?
```

Figure 22: Collecting Process Class

4.5.1. SCTP Collector Class

The SctpCollector class contains the configuration parameters of a listening SCTP socket at a collecting process.

```
     +--rw sctp-collector* [name] {sctp-transport}?
        +--rw name                        name-type
        +--rw local-port?                 inet:port-number
        |      +--rw transport-layer-security!
        |      |     ...
        +--rw (local-address-method)?
        |  +--:(local-address)
        |      +--rw local-address*        inet:host
        +--ro transport-session* [name]
            ...
```

Figure 23: SCTP Collector Class

The parameters are:

local-ip-address
   List of local IP addresses on which the collecting process listens for IPFIX messages. The IP addresses are used as eligible local IP addresses of the multihomed SCTP endpoint [RFC4960]. IF omitted, the collecting process listens on all local IP addresses.

local-port

Local port number on which the collecting process listens for IPFIX messages. If omitted, standard port 4739 (IPFIX without TLS and DTLS) or 4740 (IPFIX over TLS or DTLS) is used.

Using the TransportLayerSecurity class described in Section 4.6, DTLS is enabled and configured for this receiving socket.

The TransportSession class is specified in Section 4.7.

4.5.2. UDP Collector Class

The UdpCollector class shown in Figure 24 contains the configuration parameters of a listening UDP socket at a collecting process. The parameter local-port has the same meaning as in the SctpCollector class (see Section 4.5.1).

```
+--rw udp-collector* [name] {udp-transport}?
   +--rw name                             name-type
   +--rw local-port?                      inet:port-number
   +--rw transport-layer-security!
   |      ...
   +--rw (local-address-method)?
   |  +--:(local-address)
   |     +--rw local-address*            inet:host
   +--rw template-life-time?              uint32
   +--rw options-template-life-time?      uint32
   +--rw template-life-packet?            uint32
   +--rw options-template-life-packet?    uint32
   +--ro transport-session* [name]
      ...
```

Figure 24: UDP Collector Class

The remaining parameters are:

local-ip-address
   List of local IP addresses on which the collecting process listens for IPFIX messages. If omitted, the collecting process listens on all local IP addresses.

template-life-time, options-template-life-time
   (options) template lifetime in seconds for all UDP transport sessions terminating at this UDP socket. (options) templates that are not received again within the configured lifetime become invalid at the collecting process. As specified in [RFC7011], section 10.3.7, the lifetime of templates and options templates must be at least three times higher than the template-refresh-timeout and option-templates-refresh-timeout parameter values

configured on the corresponding exporting processes. If not configured, the default value 1800 is used, which is three times the default (options) template refresh timeout (see Section 4.4.2) as specified in [RFC7011]. Note that these parameters correspond to ipfixTransportSessionTemplateRefreshTimeout and ipfixTransportSessionOptionsTemplateRefreshTimeout in the IPFIX MIB module [RFC6615].

template-life-packet, options-template-life-packet
If template-life-packet is configured, templates defined in a UDP transport session become invalid if they are neither included in a sequence of more than this number of IPFIX messages nor received again within the period of time specified by template-lifetime. Similarly, if options-template-life-packet is configured, options templates become invalid if they are neither included in a sequence of more than this number of IPFIX messages nor received again within the period of time specified by options-template-lifetime. If not configured, templates and options templates only become invalid according to the lifetimes specified by template-lifetime and options-template-lifetime, respectively. Note that these parameters correspond to ipfixTransportSessionTemplateRefreshPacket and ipfixTransportSessionOptionsTemplateRefreshPacket in the IPFIX MIB module [RFC6615].

Using the TransportLayerSecurity class described in Section 4.6, DTLS is enabled and configured for this receiving socket.

The TransportSession class is specified in Section 4.7.

4.5.3. TCP Collector Class

The TcpCollector class contains the configuration parameters of a listening TCP socket at a collecting process. The parameters have the same meaning as in the UdpCollector class (Section 4.5.2).

Using the TransportLayerSecurity class described in Section 4.6, TLS is enabled and configured for this receiving socket.

The TransportSession class is specified in Section 4.7.

```
       +--rw tcp-collector* [name] {tcp-transport}?
          +--rw name                        name-type
          +--rw local-port?                 inet:port-number
          +--rw transport-layer-security!
          |     ...
          +--rw (local-address-method)?
          |  +--:(local-address)
          |     +--rw local-address*        inet:host
          +--ro transport-session* [name]
            ...
```

Figure 25: TCP Collector Class

4.5.4. File Reader Class

Figure 26 shows the FileReader class via which the collecting process may import IPFIX messages from a file as specified in [RFC5655].

```
       +--rw file-reader* [name] {file-reader}?
          +--rw name                 name-type
          +--rw file                 inet:uri
          +--ro file-reader-state
             +--ro bytes?                               yang:counter64
             +--ro messages?                            yang:counter64
             +--ro records?                             yang:counter64
             +--ro templates?                           yang:counter32
             +--ro options-templates?                   yang:counter32
             +--ro file-reader-discontinuity-time?
             |       yang:date-and-time
             +--ro template* []
                ...
```

Figure 26: File Reader Class

The FileReader class defines the following configuration parameter:

file
   File name and location specified as URI.

The state parameters of the FileReader class are:

bytes, messages, records, templates, options-templates
   The number of bytes, IPFIX messages, data records, template records, and options template records read by the file reader. Discontinuities in the values of these counters can occur at re-initialization of the management system, and at other times as indicated by the value of file-reader-discontinuity-time.

file-reader-discontinuity-time
Timestamp of the most recent occasion at which one or more file reader counters suffered a discontinuity. The time is absolute and not relative to sysUpTime.

The FileReader class includes information about the Template class and statistics. The Template class is specified in Section 4.8.

4.6. Transport Layer Security Class

Figure 27 shows the TransportLayerSecurity class which is used in the exporting process's sctp-exporter, udp-exporter, and TcpExporter classes, and the collecting process's SctpCollector, UdpCollector, and TcpCollector classes to enable and configure TLS/DTLS for IPFIX. If TLS/DTLS is enabled, the endpoint must use DTLS [RFC6347] if the transport protocol is SCTP or UDP and TLS [RFC8446] if the transport protocol is TCP.

[RFC7011] mandates strong mutual authentication of exporting processes and collecting process as follows. IPFIX exporting processes and IPFIX collecting processes are identified by the fully qualified domain name (FQDN) of the interface on which IPFIX messages are sent or received, for purposes of X.509 client and server certificates as in [RFC5280]. To prevent man-in-the-middle attacks from impostor exporting or collecting processes, the acceptance of data from an unauthorized exporting process, or the export of data to an unauthorized collecting process, strong mutual authentication via asymmetric keys must be used for both TLS and DTLS. Each of the IPFIX exporting and collecting processes must verify the identity of its peer against its authorized certificates, and must verify that the peer's certificate matches its fully qualified domain name, or, in the case of SCTP, the fully qualified domain name of one of its endpoints.

The fully qualified domain name used to identify an IPFIX collecting process or exporting process may be stored either in a subjectaltname extension of type dnsname, or in the most specific common name field of the subject field of the x.509 certificate. If both are present, the subjectaltname extension is given preference.

In order to use TLS/DTLS, appropriate certificates and keys have to be previously installed on the monitoring devices. For security reasons, the configuration data model does not offer the possibility to upload any certificates or keys on a monitoring device. If TLS/DTLS is enabled on a monitoring device that does not dispose of appropriate certificates and keys, the configuration must be rejected with an error.

The configuration data model allows restricting the authorization of remote endpoints to certificates issued by specific certification authorities or identifying specific fqdns for authorization. Furthermore, the configuration data model allows restricting the utilization of certificates identifying the local endpoint. This is useful if the monitoring device disposes of more than one certificate for the given local endpoint.

```
     +--rw transport-layer-security!
        +--rw local-certification-authority-dn*    string
        +--rw local-subject-dn*                    string
        +--rw local-subject-fqdn*                  inet:domain-name
        +--rw remote-certification-authority-dn*   string
        +--rw remote-subject-dn*                   string
        +--rw remote-subject-fqdn*                 inet:domain-name
```

Figure 27: Transport Layer Security Class

The configuration parameters are defined as follows:

local-certification-authority-dn
This parameter may appear one or more times to restrict the identification of the local endpoint during the tls/dtls handshake to certificates issued by the configured certification authorities. each occurrence of this parameter contains the distinguished name of one certification authority. To identify the local endpoint, the exporting process or collecting process must use a certificate issued by one of the configured certification authorities. Certificates issued by any other certification authority must not be sent to the remote peer during TLS/DTLS handshake. If none of the certificates installed on the monitoring device fulfills the specified restrictions, the configuration must be rejected with an error. If local-certification-authority-dn is not configured, the choice of certificates identifying the local endpoint is not restricted with respect to the issuing certification authority.

local-subject-dn, local-subject-fqdn
Each of these parameters may appear one or more times to restrict the identification of the local endpoint during the TLS/DTLS handshake to certificates issued for specific subjects or for specific FQDNs. Each occurrence of local-subject-dn contains a distinguished name identifying the local endpoint. Each occurrence of local-subject-fqdn contains a FQDN which is assigned to the local endpoint. To identify the local endpoint, the exporting process or collecting process must use a certificate that contains either one of the configured distinguished names in the subject field or at least one of the configured FQDNs in a

dnsname component of the subject alternative extension field or in the most specific commonname component of the subject field. If none of the certificates installed on the monitoring device fulfills the specified restrictions, the configuration must be rejected with an error. If any of the parameters local-subject-dn and local-subject-fqdn is configured at the same time as the local-certification-authority-dn parameter, certificates must also fulfill the specified restrictions regarding the certification authority. If local-subject-dn and local-subject-fqdn are not configured, the choice of certificates identifying the local endpoint is not restricted with respect to the subject's distinguished name or FQDN.

remote-certification-authority-dn
This parameter may appear one or more times to restrict the authentication of remote endpoints during the TLS/DTLS handshake to certificates issued by the configured certification authorities. Each occurrence of this parameter contains the distinguished name of one certification authority. To authenticate the remote endpoint, the remote exporting process or collecting process must provide a certificate issued by one of the configured certification authorities. Certificates issued by any other certification authority must be rejected during TLS/DTLS handshake. If the monitoring device is not able to validate certificates issued by the configured certification authorities (e.g., because of missing public keys), the configuration must be rejected with an error. If remote-certification-authority-dn is not configured, the authorization of remote endpoints is not restricted with respect to the issuing certification authority of the delivered certificate.

remote-subject-dn, remote-subject-fqdn
Each of these parameters may appear one or more times to restrict the authentication of remote endpoints during the TLS/DTLS handshake to certificates issued for specific subjects or for specific FQDNs. Each occurrence of remote-subject-dn contains a distinguished name identifying a remote endpoint. Each occurrence of remote-subject-fqdn contains a FQDN that is assigned to a remote endpoint. To authenticate a remote endpoint, the remote exporting process or collecting process must provide a certificate that contains either one of the configured distinguished names in the subject field or at least one of the configured FQDNs in a dnsname component of the subject alternative extension field or in the most specific common name component of the subject field. Certificates not fulfilling this condition must be rejected during TLS/DTLS handshake. If any of the parameters remote-subject-dn and remote-subject-fqdn is configured at the same time as the remote-certification-authority-dn parameter, certificates must

also fulfill the specified restrictions regarding the certification authority in order to be accepted. If remote-subject-dn and remote-subject-FQDN are not configured, the authorization of remote endpoints is not restricted with respect to the subject's distinguished name or FQDN of the delivered certificate.

4.7. Transport Session Class

The TransportSession class contains state data about transport sessions originating from an exporting process or terminating at a collecting process. If SCTP is the transport protocol, the exporter or collector may be multihomed SCTP endpoints (see [RFC4960], Section 6.4), in which case more than one IP address will be used.

The following attributes are supported:

ipfix-version
: Used for exporting processes, this parameter contains the version number of the IPFIX protocol that the exporter uses to export its data in this transport session. Hence, it is identical to the value of the configuration parameter ipfix-version of the sctp-exporter, udp-exporter, or tcp-exporter object. When used for collecting processes, this parameter contains the version-number of the IPFIX protocol it receives for this transport session. If IPFIX messages of different IPFIX protocol versions are received, this parameter contains the maximum version number. This state parameter is identical to ipfixTransportSessionIpfixVersion in the IPFIX MIB module [RFC6615].

source-address, destination-address
: If TCP or UDP is the transport protocol, source-address contains the IP address of the exporter, and destination-address contains the IP addresses of the collector. Hence, the two parameters have identical values as ipfixTransportSessionSourceAddress and ipfixTransportSessionDestinationAddress in the IPFIX MIB module [RFC6615]. if SCTP is the transport protocol, source-address contains one of the IP addresses of the exporter and destination-address one of the IP addresses of the collector. Preferably, the IP addresses of the path that is usually selected by the exporter to send IPFIX messages to the collector should be contained.

source-port, destination-port
: These state parameters contain the transport-protocol port numbers of the exporter and the collector of the transport session and thus are identical to ipfixTransportSessionSourcePort and ipfixTransportSessionDestinationPort in the IPFIX MIB module [RFC6615].

sctp-assoc-id
The association id used for the SCTP session between the exporter and the collector of the transport session. It is equal to the sctpassocid entry in the SctpAssocTable defined in the SCTP-MIB [RFC3871]. This parameter is only available if the transport protocol is SCTP and if an SNMP agent on the same monitoring device enables access to the corresponding MIB objects in the SctpAssocTable. This state parameter is identical to ipfixTransportSessionSctpAssocId in the IPFIX MIB module [RFC6615].

status
Status of the transport session, which can be one of the following:

* inactive: transport session is established, but no IPFIX messages are currently transferred (e.g., because this is a backup (secondary) session)

* active: transport session is established and transfers IPFIX messages

* unknown: transport session status cannot be determined; this state parameter is identical to ipfixTransportSessionStatus in the IPFIX MIB module [RFC6615]

rate
The number of bytes per second transmitted by the exporting process or received by the collecting process. This parameter is updated every second. This state parameter is identical to ipfixtransportsessionrate in the IPFIX MIB module [RFC6615].

bytes, messages, records, templates, options-templates
The number of bytes, IPFIX messages, data records, template records, and options template records transmitted by the exporting process or received by the collecting process. Discontinuities in the values of these counters can occur at re-initialization of the management system, and at other times as indicated by the value of transport-session-discontinuity-time.

discarded-messages
Used for exporting processes, this parameter indicates the number of messages that could not be sent due to internal buffer overflows, network congestion, routing issues, etc. Used for collecting process, this parameter indicates the number of received IPFIX messages that are malformed, cannot be decoded, are received in the wrong order or are missing according to the sequence number. Discontinuities in the value of this counter can

occur at re-initialization of the management system, and at other times as indicated by the value of transport-session-discontinuity-time.

transport-session-start-time
Timestamp of the start of the given transport session.

transport-session-discontinuity-time
Timestamp of the most recent occasion at which one or more of the transport session counters suffered a discontinuity. The time is absolute and not relative to sysUpTime. Note that, if used for exporting processes, the values of the state parameters destination-address and destination-port match the values of the configuration parameters destination-ip-address and destination-port of the sctp-exporter, tcp-exporter, and udp-exporter (in the case of sctp-exporter, one of the configured destination-ip-address values); if the transport protocol is UDP or SCTP and if the parameter source-ip-address is configured in the udp-exporter or sctp-exporter object, the value of source-address equals the configured value or one of the configured values. Used for collecting processes, the value of destination-address equals the value (or one of the values) of the parameter local-ip-address if this parameter is configured in the udp-collector, tcp-collector, or sctp-collector; destination-port equals the value of the configuration parameter local-port.

The TransportSession class includes Template class information and statistics about the templates transmitted or received on the given transport session. The Template class is specified in Section 4.8.

```
        +--ro transport-session* [name]
           +--ro name                                        name-type
           +--ro ipfix-version?                              uint16
           +--ro source-address?                             inet:host
           +--ro destination-address?                        inet:host
           +--ro source-port?
           |       inet:port-number
           +--ro destination-port?
           |       inet:port-number
           +--ro status?
           |       transport-session-status
           +--ro rate?
           |       yang:gauge32
           +--ro bytes?
           |       yang:counter64
           +--ro messages?
           |       yang:counter64
           +--ro discarded-messages?
           |       yang:counter64
           +--ro records?
           |       yang:counter64
           +--ro templates?
           |       yang:counter32
           +--ro options-templates?
           |       yang:counter32
           +--ro transport-session-start-time?
           |       yang:date-and-time
           +--ro transport-session-discontinuity-time?
           |       yang:date-and-time
           +--ro template* []
             ...
```

Figure 28: Transport Session Class

4.8. Template Class

Figure 29 shows the Template class which contains state data about templates used by an exporting process or received by a collecting process in a specific transport session. The field class defines one field of the template.

```
        +--ro template* []
           +--ro observation-domain-id?         uint32
           +--ro template-id?                   uint16
           +--ro set-id?                        uint16
           +--ro access-time?                   yang:date-and-time
           +--ro template-data-records?         yang:counter64
           +--ro template-discontinuity-time?   yang:date-and-time
           +--ro field* []
              +--ro ie-id?                  ie-id-type
              +--ro ie-length?              uint16
              +--ro ie-enterprise-number?   uint32
              +--ro is-flow-key?            empty
              +--ro is-scope?               empty
```

Figure 29: Template Class

The names and semantics of the state parameters correspond to the managed objects in the ipfixTemplateTable, ipfixTemplateDefinitionTable, and ipfixTemplateStatsTable of the IPFIX MIB module [RFC6615]:

observation-domain-id
The identifier of the observation domain for which this template is defined.

template-id
This number indicates the template identifier in the IPFIX Message.

set-id
This number indicates the set identifier of this template. Currently, there are two values defined [RFC7011]. The value 2 is used for sets containing template definitions. The value 3 is used for sets containing options template definitions.

access-time
Used for exporting processes, this parameter contains the time when this (Options) Template was last sent to the Collector or written to the file. Used for Collecting Processes, this parameter contains the time when this (Options) Template was last received from the Exporter or read from the file.

template-data-records
The number of transmitted or received data records defined by this (options) template since the point in time indicated by template-definition-time.

template-discontinuity-time

Timestamp of the most recent occasion at which the counter template-data-records suffered a discontinuity. The time is absolute and not relative to sysUpTime.

ie-id, ie-length, ie-enterprise-number

Information Element identifier, length, and enterprise number of a field in the template. If this is not an enterprise-specific Information Element, ie-enterprise-number is zero. These state parameters are identical to ipfixTemplateDefinitionIeId, ipfixTemplateDefinitionIeLength, and ipfixTemplateDefinitionIeEnterpriseNumber in the IPFIX MIB module [RFC6615].

is-flow-key

If this state parameter is present, this is a flow key field. This parameter is only available for non-Options Templates (i.e., if setId is 2).

is-scope

If this state parameter is present, this is a scope field. This parameter is only available for options templates (i.e., if setId is 3).

4.9. Bulk Data Class

The BulkDataProcess class in Figure 30 specifies the bulk data template to be applied to resource or set of resources and provides state information about the template records.

```
        +--rw bulk-data-export
           +--rw template* [name]
              +--rw name                        ietf-ipfix:name-type
              +--rw enabled?                    boolean
              +--rw export-interval?            uint32
              +--rw observation-domain-id?      uint32
              +--rw field-layout
              |  +--rw field* [name]
              |     +--rw name                       ietf-ipfix:name-type
              |     +--rw (identifier)
              |     |  +--:(ie-id)
              |     |     +--rw ie-id?               ietf-ipfix:ie-id-type
              |     +--rw ie-length?                 uint16
              |     +--rw ie-enterprise-number?      uint32
              +--rw exporting-process*
              |       -> /ietf-ipfix:ipfix/exporting-process/name
              |       {ietf-ipfix:exporter}?
              +--rw resource*                   resource
              +--ro data-records?               yang:counter64
              +--ro discontinuity-time?         yang:date-and-time
```

Figure 30: Bulk Data Class

The following attributes are supported:

enabled
   Enables the template so that specified data may be exported. The default is "enabled".

export-interval
   The interval (in seconds) for periodical export of data records.

observation-domain-id
   The Observation Domain that is locally unique to an Exporting Process

field-layout
   The IPFIX template to be applied to the resource. The following attributes are configurable:

   * ie-id: Identifies the Information Element identifier.

   * ie-enterprise-id: Identifies the enterprise identifier of the Information Element. If 0, the enterprise ID is an IANA based Information Element.

   * ie-length: Identifies the length of the Information Element.

A bulk data instance may refer to:

- o one or more exporting-process instances

- o one or more resource instances (e.g., different interface instances on a line card)

The following state information is available;

data-records
: Reports the number of data records generated for this bulk data template.

discontinuity-time
: Timestamp of the most recent occasion at which the counter data records suffered a discontinuity.

5. Adaptation to Device Capabilities

The configuration data model standardizes a superset of common IPFIX and PSAMP configuration parameters. A typical monitoring device implementation will not support the entire range of possible configurations. Certain functions may not be supported, such as the collecting process that does not exist on a monitoring device that is conceived as exporter only. The configuration of other functions may be subject to resource limitations or functional restrictions. For example, the cache size is typically limited according to the available memory on the device. It is also possible that a monitoring device implementation requires the configuration of additional parameters that are not part of the configuration data model in order to function properly.

The configuration data model for IPFIX and PSAMP covers the configuration of Exporters, Collectors, and devices that may act as both. As Exporters and Collectors implement different functions, the corresponding portions of the model are conditional on the following features:

exporter
: If this feature is supported, Exporting Processes can be configured.

collector
: If this feature is supported, Collecting Processes can be configured.

Exporters do not necessarily implement any Selection Processes, Caches, or even Observation Points in particular cases. Therefore, the corresponding portions of the model are conditional on the following feature:

Additional features refer to different PSAMP Sampling and Filtering methods as well as to the supported types of Caches:

psamp-samp-count-based
   If this feature is supported, Sampling method samp-count-based can be configured.

psamp-samp-time-based
   If this feature is supported, Sampling method samp-time-based can be configured.

psamp-samp-rand-out-of-n
   If this feature is supported, Sampling method samp-rand-out-of-n can be configured.

psamp-samp-uni-prob
   If this feature is supported, Sampling method samp-uni-prob can be configured.

psampfilter-match
   If this feature is supported, Filtering method filter-match can be configured.

psamp-filter-hash
   If this feature is supported, Filtering method filter-hash can be configured.

immediate-cache
   If this feature is supported, a Cache generating PSAMP Packet Reports can be configured using the Immediate Cache class.

timeout-cache
   If this feature is supported, a Cache generating IPFIX Flow Records can be configured using the Timeout Cache class.

natural-cache
   If this feature is supported, a Cache generating IPFIX Flow Records can be configured using the Natural Cache class.

permanent-cache
   If this feature is supported, a Cache generating IPFIX Flow Records can be configured using the Permanent Cache class.

The following features concern the support of UDP and TCP as transport protocols and the support of File Readers and File Writers:

sctp-transport
If this feature is supported, SCTP can be used as transport protocol by Exporting Processes and Collecting Processes.

udp-transport
If this feature is supported, UDP can be used as transport protocol by Exporting Processes and Collecting Processes.

tcp-transport
If this feature is supported, TCP can be used as transport protocol by Exporting Processes and Collecting Processes.

file-reader
If this feature is supported, File Readers can be configured as part of Collecting Processes.

file-writer
If this feature is supported, File Writers can be configured as part of Exporting Processes.

6. YANG Modules

This document defines three YANG modules:

ietf-ipfix
Defines the IPFIX collector and exporter functions.

ietf-ipfix-packet-sampling
Defines the PSAMP functions for configuring a device to sample/ meter a subset of packets from the network.

ietf-ipfix-bulk-data-export
Defines the bulk data IPFIX templates used to export bulk data.

6.1. ietf-ipfix

6.1.1. ietf-ipfix Module Structure

This document defines the YANG module "ietf-ipfix", which has the following structure:

```
   module: ietf-ipfix
     +--rw ipfix
        +--rw collecting-process* [name] {collector}?
        |  +--rw name                    name-type
        |  +--rw tcp-collector* [name] {tcp-transport}?
        |  |      ...
        |  +--rw udp-collector* [name] {udp-transport}?
        |  |      ...
        |  +--rw sctp-collector* [name] {sctp-transport}?
        |  |      ...
        |  +--rw file-reader* [name] {file-reader}?
        |  |      ...
        |  +--rw exporting-process*   -> /ipfix/exporting-process/name
        |          {exporter}?
        +--rw exporting-process* [name] {exporter}?
           +--rw name                    name-type
           +--rw enabled?                boolean
           +--rw export-mode?            identityref
           +--rw destination* [name]
           |      ...
           +--rw options* [name]
           |      ...
           +--ro exporting-process-id?   uint32
```

6.1.2. ietf-ipfix YANG Module

This YANG Module imports typedefs from [RFC6991].

<CODE BEGINS> file "ietf-ipfix@2018-10-22.yang"

```
   module ietf-ipfix {
     yang-version 1.1;

     namespace "urn:ietf:params:xml:ns:yang:ietf-ipfix";

     prefix ietf-ipfix;

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

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

```
     import ietf-interfaces {
       prefix if;
       reference
         "RFC 8343: A YANG Model for Interface Management";
     }

     organization
       "IETF";

     contact
       "Web:      TBD
        List:     TBD

        Editor:   Joey Boyd
                  <mailto:joey.boyd@adtran.com>

        Editor:   Marta Seda
                  <mailto:marta.seda@calix.com>";

     // RFC Ed.: replace XXXX with actual RFC numbers and
     // remove this note.

     description
       "This module contains a collection of YANG definitions for the
        management of IP Flow Information Export (IPFIX).

        This data model is designed for the Network Management Datastore
        Architecture defined in RFC 8342.

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

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

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

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

```
     revision 2020-03-05 {
       description
         "Initial revision.";
       reference
         "RFC XXXX: YANG Data Models for the IP Flow Information Export
                    (IPFIX) Protocol, Packet Sampling (PSAMP) Protocol,
                    and Bulk Data Export";
     }

     feature exporter {
       description
         "If supported, the Monitoring Device can be used as
          an Exporter. Exporting Processes can be configured.";
     }

     feature collector {
       description
         "If supported, the Monitoring Device can be used as
          a Collector.  Collecting Processes can be configured.";
     }

     feature tcp-transport {
       description
         "If supported, the Monitoring Device supports TCP
          as the transport protocol.";
     }

     feature udp-transport {
       description
         "If supported, the Monitoring Device supports UDP
          as the transport protocol.";
     }

     feature sctp-transport {
       description
         "If supported, the Monitoring Device supports SCTP
          as the transport protocol.";
     }

     feature file-reader {
       description
         "If supported, the Monitoring Device supports the
          configuration of Collecting Processes as File Readers.";
     }

     feature file-writer {
       description
         "If supported, the Monitoring Device supports the
```

```
          configuration of Exporting Processes as File Writers.";
     }

     feature if-mib {
       description
         "This feature indicates that the device implements
          the IF-MIB.";
       reference
         "RFC 2863: The Interfaces Group MIB";
     }

     identity export-mode {
       description
         "Base identity for different usages of export
          destinations configured for an Exporting Process.";
       reference
         "RFC 6615, Section 8 (ipfixExportMemberType)";
     }

     identity parallel {
       base export-mode;
       description
         "Parallel export of Data Records to all destinations configured
          for the Exporting Process.";
       reference
         "RFC 6615, Section 8 (ipfixExportMemberType)";
     }

     identity load-balancing {
       base export-mode;
       description
         "Load-balancing between the different destinations
          configured for the Exporting Process.";
       reference
         "RFC 6615, Section 8 (ipfixExportMemberType)";
     }

     identity fallback {
       base export-mode;
       description
         "Export to the primary destination (i.e., the first
          destination configured for the Exporting Process). If the
          export to the primary destination fails, the Exporting Process
          tries to export to the secondary destination.  If the
          secondary destination fails as well, it continues with the
          tertiary, etc.";
       reference
         "RFC 6615, Section 8 (ipfixExportMemberType)";
```

```
     }

     identity options-type {
       description
         "Base identity for report types exported with
          options templates.";
     }

     identity metering-statistics {
       base options-type;
       description
         "Metering Process Statistics.";
       reference
         "RFC 7011, Section 4.1";
     }

     identity metering-reliability {
       base options-type;
       description
         "Metering Process Reliability Statistics.";
       reference
         "RFC 7011, Section 4.2";
     }

     identity exporting-reliability {
       base options-type;
       description
         "Exporting Process Reliability Statistics.";
       reference
         "RFC 7011, Section 4.3";
     }

     identity flow-keys {
       base options-type;
       description
         "Flow Keys.";
       reference
         "RFC 7011, Section 4.4";
     }

     identity selection-sequence {
       base options-type;
       description
         "Selection Sequence and Selector Reports.";
       reference
         "RFC 5476, Sections 6.5.1 and 6.5.2";
     }
```

```
     identity selection-statistics {
       base options-type;
       description
         "Selection Sequence Statistics Report.";
       reference
         "RFC 5476, Sections 6.5.3";
     }

     identity accuracy {
       base options-type;
       description
         "Accuracy Report.";
       reference
         "RFC 5476, Section 6.5.4";
     }

     identity reducing-redundancy {
       base options-type;
       description
         "Enables the utilization of Options Templates to reduce
          redundancy in the exported Data Records.";
       reference
         "RFC 5473";
     }

     identity extended-type-information {
       base options-type;
       description
         "Export of extended type information for enterprise-specific
          Information Elements used in the exported Templates.";
       reference
         "RFC 5610";
     }

     typedef ie-name-type {
       type string {
         length "1..max";
         pattern '\S+';
       }
       description
         "Type for Information Element names. Whitespaces are not
          allowed.";
     }

     typedef name-type {
       type string {
         length "1..max";
         pattern '\S(.*\S)?';
```

```
       }
       description
         "Type for 'name' leafs, which are used to identify specific
          instances within lists, etc.

          Leading and trailing whitespaces are not allowed.";
     }

     typedef ie-id-type {
       type uint16 {
         range "1..32767";
       }
       description
         "Type for Information Element identifiers.";
     }

     typedef transport-session-status {
       type enumeration {
         enum "inactive" {
           value 0;
           description
             "This value MUST be used for Transport Sessions that are
              specified in the system but currently not active.

              The value can be used for Transport Sessions that are
              backup (secondary) sessions.";
         }
         enum "active" {
           value 1;
           description
             "This value MUST be used for Transport Sessions that are
              currently active and transmitting or receiving data.";
         }
         enum "unknown" {
           value 2;
           description
             "This value MUST be used if the status of the Transport
              Sessions cannot be detected by the device.

              This value should be avoided as far as possible.";
         }
       }
       description
         "Status of a Transport Session.";
       reference
         "RFC 6615, Section 8 (ipfixTransportSessionStatus)";
     }
```

```
     grouping transport-layer-security-parameters {
       description
         "TLS or DTLS parameters.";

       container transport-layer-security {
         presence
           "The presence of this container indicates TLS is enabled.";
         description
           "TLS or DTLS configuration.";

         leaf-list local-certification-authority-dn {
           type string;
           description
             "Distinguished names of certification authorities whose
              certificates may be used to identify the local endpoint.";
           reference
             "RFC 5280";
         }

         leaf-list local-subject-dn {
           type string;
           description
             "Distinguished names that may be used in the certificates
              to identify the local endpoint.";
           reference
             "RFC 5280.";
         }

         leaf-list local-subject-fqdn {
           type inet:domain-name;
           description
             "Fully qualified domain names that may be used in the
              certificates to identify the local endpoint.";
           reference
             "RFC 5280";
         }

         leaf-list remote-certification-authority-dn {
           type string;
           description
             "Distinguished names of certification authorities whose
              certificates are accepted to authorize remote endpoints.";
           reference
             "RFC 5280";
         }

         leaf-list remote-subject-dn {
           type string;
```

```
          description
            "Distinguished names which are accepted in certificates to
             authorize remote endpoints.";
          reference
            "RFC 5280";
        }

        leaf-list remote-subject-fqdn {
          type inet:domain-name;
          description
            "Fully qualified domain names that are accepted in
             certificates to authorize remote endpoints.";
          reference
            "RFC 5280";
        }
      }
    }

    grouping transport-session-state-parameters {
      description
        "State parameters of a Transport Session originating from an
         Exporting Process or terminating at a Collecting Process.
         Parameter names and semantics correspond to the managed
         objects in IPFIX-MIB.";
      reference
        "RFC 7011; RFC 6615, Section 8 (ipfixTransportSessionEntry,
         ipfixTransportSessionStatsEntry)";

      leaf ipfix-version {
        type uint16;
        description
          "Used for Exporting Processes, this parameter contains the
           version number of the IPFIX protocol that the Exporter uses
           to export its data in this Transport Session.

           Used for Collecting Processes, this parameter contains the
           version number of the IPFIX protocol it receives for this
           Transport Session. If IPFIX Messages of different IPFIX
           protocol versions are received, this parameter contains the
           maximum version number.

           Note that this parameter corresponds to
           ipfixTransportSessionIpfixVersion in the IPFIX MIB module.";
        reference
          "RFC 6615, Section 8
           (ipfixTransportSessionIpfixVersion)";
      }
```

```
       leaf source-address {
         type inet:host;
         description
           "The source address of the Exporter of the IPFIX Transport
            Session.";
         reference
           "RFC 6615, Section 8
            (ipfixTransportSessionSourceAddressType,
            ipfixTransportSessionSourceAddress);
            RFC 4960, Section 6.4";
       }

       leaf destination-address {
         type inet:host;
         description
           "The destination address of the path that is selected by the
            Exporter to send IPFIX messages to the Collector.

            In the case of TCP, it is possible that if an FQDN address
            is configured it resolves into many addresses.

            Note that this parameter functionally corresponds to
            ipfixTransportSessionDestinationAddressType and
            ipfixTransportSessionDestinationAddress in the IPFIX MIB
            module.";
         reference
           "RFC 6615, Section 8
            (ipfixTransportSessionDestinationAddressType,
            ipfixTransportSessionDestinationAddress);
            RFC 4960, Section 6.4";
       }

       leaf source-port {
         type inet:port-number;
         description
           "The transport-protocol port number of the Exporter of the
            IPFIX Transport Session.

            Note that this parameter corresponds to
            ipfixTransportSessionSourcePort in the IPFIX MIB module.";
         reference
           "RFC 6615, Section 8
            (ipfixTransportSessionSourcePort).";
       }

       leaf destination-port {
         type inet:port-number;
         description
```

```
            "The transport-protocol port number of the Collector of the
             IPFIX Transport Session.

             Note that this parameter corresponds to
             ipfixTransportSessionDestinationPort in the IPFIX MIB
             module.";
          reference
            "RFC 6615, Section 8
             (ipfixTransportSessionDestinationPort)";
        }

        leaf status {
          type transport-session-status;
          description
            "Status of the Transport Session.

             Note that this parameter corresponds to
             ipfixTransportSessionStatus in the IPFIX MIB module.";
            reference
             "RFC 6615, Section 8 (ipfixTransportSessionStatus)";
        }

        leaf rate {
          type yang:gauge32;
          units "bytes per second";
          description
            "The number of bytes per second transmitted by the
             Exporting Process or received by the Collecting Process.
             This parameter is updated every second.

             Note that this parameter corresponds to
             ipfixTransportSessionRate in the IPFIX MIB module.";
          reference
            "RFC 6615, Section 8 (ipfixTransportSessionRate)";
        }

        leaf bytes {
          type yang:counter64;
          units "bytes";
          description
            "The number of bytes transmitted by the Exporting Process or
             received by the Collecting Process.

             Discontinuities in the value of this counter can occur at
             re-initialization of the management system, and at other
             times as indicated by the value of
             transport-session-discontinuity-time.
```

```
            Note that this parameter corresponds to
            ipfixTransportSessionBytes in the IPFIX MIB module.";
         reference
           "RFC 6615, Section 8 (ipfixTransportSessionBytes)";
       }

       leaf messages {
          type yang:counter64;
          units "IPFIX Messages";
          description
            "The number of messages transmitted by the Exporting Process
             or received by the Collecting Process.

             Discontinuities in the value of this counter can occur at
             re-initialization of the management system, and at other
             times as indicated by the value of
             transport-session-discontinuity-time.

             Note that this parameter corresponds to
             ipfixTransportSessionMessages in the IPFIX MIB module.";
          reference
            "RFC 6615, Section 8
             (ipfixTransportSessionMessages)";
       }

       leaf discarded-messages {
         type yang:counter64;
         units "IPFIX Messages";
         description
           "Used for Exporting Processes, this parameter indicates the
            number of messages that could not be sent due to internal
            buffer overflows, network congestion, routing issues, etc.
            Used for Collecting Process, this parameter indicates the
            number of received IPFIX Message that are malformed, cannot
            be decoded, are received in the wrong order or are missing
            according to the sequence number.

            Discontinuities in the value of this counter can occur at
            re-initialization of the management system, and at other
            times as indicated by the value of
            transport-session-discontinuity-time.

            Note that this parameter corresponds to
            ipfixTransportSessionDiscardedMessages in the IPFIX MIB
            module.";
         reference
           "RFC 6615, Section 8
            (ipfixTransportSessionDiscardedMessages)";
```

```
       }

       leaf records {
         type yang:counter64;
         units "Data Records";
         description
           "The number of Data Records transmitted by the Exporting
            Process or received by the Collecting Process.

            Discontinuities in the value of this counter can occur at
            re-initialization of the management system, and at other
            times as indicated by the value of
            transport-session-discontinuity-time.

            Note that this parameter corresponds to
            ipfixTransportSessionRecords in the IPFIX MIB module.";
         reference
           "RFC 6615, Section 8
            (ipfixTransportSessionRecords)";
       }

       leaf templates {
         type yang:counter32;
         units "Templates";
         description
           "The number of Templates transmitted by the Exporting Process
            or received by the Collecting Process.

            Discontinuities in the value of this counter can occur at
            re-initialization of the management system, and at other
            times as indicated by the value of
            transport-session-discontinuity-time.

            Note that this parameter corresponds to
            ipfixTransportSessionTemplates in the IPFIX MIB module.";
         reference
           "RFC 6615, Section 8
           (ipfixTransportSessionTemplates)";
       }

       leaf options-templates {
         type yang:counter32;
         units "Options Templates";
         description
           "The number of Option Templates transmitted by the Exporting
            Process or received by the Collecting Process.

            Discontinuities in the value of this counter can occur at
```

```
            re-initialization of the management system, and at other
            times as indicated by the value of
            transport-session-discontinuity-time.

            Note that this parameter corresponds to
            ipfixTransportSessionOptionsTemplates in the IPFIX MIB
            module.";
         reference
           "RFC 6615, Section 8
            (ipfixTransportSessionOptionsTemplates)";
       }

       leaf transport-session-start-time {
         type yang:date-and-time;
         description
           "Timestamp of the start of the given Transport Session.

            This state parameter does not correspond to any object in
            the IPFIX MIB module.";
       }

       leaf transport-session-discontinuity-time {
         type yang:date-and-time;
         description
           "Timestamp of the most recent occasion at which one or more
            of the Transport Session counters suffered a discontinuity.

            Note that this parameter functionally corresponds to
            ipfixTransportSessionDiscontinuityTime in the IPFIX MIB
            module. In contrast to
            ipfixTransportSessionDiscontinuityTime, the time is
            absolute and not relative to sysUpTime.";
         reference
           "RFC 6615, Section 8
            (ipfixTransportSessionDiscontinuityTime)";
       }
     }

     grouping collection-template-state-parameters {
       description
         "State parameters of a (Options) Template received by a
          Collecting Process in a specific Transport Session or read by
          the File Reader.

          Parameter names and semantics correspond to the
          managed objects in IPFIX-MIB";
       reference
         "RFC 7011; RFC 6615, Section 8 (ipfixTemplateEntry,
```

```
           ipfixTemplateDefinitionEntry, ipfixTemplateStatsEntry)";

       list template {
         key "name";
         description
           "This list contains the Templates and Options Templates that
            are transmitted by the Exporting Process or received by the
            Collecting Process.

            Withdrawn or invalidated (Options) Templates MUST be removed
            from this list.";

         leaf name {
           type name-type;
           description
             "An arbitrary string which uniquely identifies the
              template.";
         }

         leaf observation-domain-id {
           type uint32;
           description
             "The ID of the Observation Domain for which this Template
              is defined.

              Note that this parameter corresponds to
              ipfixTemplateObservationDomainId in the IPFIX MIB
              module.";
           reference
             "RFC 6615, Section 8
              (ipfixTemplateObservationDomainId)";
         }

         leaf template-id {
           type uint16 {
             range "256..65535";
           }
           description
             "This number indicates the Template ID in the IPFIX
              message.

              Note that this parameter corresponds to ipfixTemplateId in
              the IPFIX MIB module.";
           reference
             "RFC 6615, Section 8 (ipfixTemplateId)";
         }

         leaf set-id {
```

```
            type uint16 {
              range "2..3 | 256..65535";
            }
            description
              "This number indicates the Set ID of the Template.
               A value of 2 is reserved for Template Sets.  A value of 3
               is reserved for Options Template Sets.  Values from 4 to
               255 are reserved for future use.  Values 256 and above
               are used for Data Sets.  The Set ID values of 0 and 1 are
               not used for historical reasons.

               Note that this parameter corresponds to ipfixTemplateSetId
               in the IPFIX MIB module.";
            reference
              "RFC 7011, Section 3.3.2;
               RFC 6615, Section 8 (ipfixTemplateSetId)";
          }

          leaf access-time {
            type yang:date-and-time;
            description
              "This parameter contains the time when this (Options)
               Template was last received from the Exporter or read from
               the file.

               Note that this parameter corresponds to
               ipfixTemplateAccessTime in the IPFIX MIB module.";
            reference
              "RFC 6615, Section 8 (
               ipfixTemplateAccessTime)";
          }

          leaf template-data-records {
            type yang:counter64;
            description
              "The number of received Data Records defined by this
               (Options) Template.

               Discontinuities in the value of this counter can occur at
               re-initialization of the management system, and at other
               times as indicated by the value of
               template-discontinuity-time.

               Note that this parameter corresponds to
               ipfixTemplateDataRecords in the IPFIX MIB module.";
            reference
              "RFC 6615, Section 8 (ipfixTemplateDataRecords)";
          }
```

```
        leaf template-discontinuity-time {
          type yang:date-and-time;
          description
            "Timestamp of the most recent occasion at which the counter
             template-data-records suffered a discontinuity.

             Note that this parameter functionally corresponds to
             ipfixTemplateDiscontinuityTime in the IPFIX MIB module. In
             contrast to  ipfixTemplateDiscontinuityTime, the time is
             absolute and not relative to sysUpTime.";
          reference
            "RFC 6615, Section 8
             (ipfixTemplateDiscontinuityTime)";
        }

        list field {
          key "name";
          description
            "This list contains the (Options) Template fields of which
             the (Options) Template is defined.

             The order of the list corresponds to the order of the
             fields in the (Option) Template Record.";

          leaf name {
            type name-type;
            description
              "An arbitrary string which uniquely identifies the
               template field.";
          }

          leaf ie-id {
            type ie-id-type;
            description
              "This parameter indicates the Information Element
               identifier of the field.

               Note that this parameter corresponds to
               ipfixTemplateDefinitionIeId in the IPFIX MIB module.";
            reference
              "RFC 7011; RFC 6615, Section 8
               (ipfixTemplateDefinitionIeId).";
          }

          leaf ie-length {
            type uint16;
            units "octets";
            description
```

```
              "This parameter indicates the length of the Information
               Element of the field.

               Note that this parameter corresponds to
               ipfixTemplateDefinitionIeLength in the IPFIX MIB
               module.";
            reference
              "RFC 7011; RFC 6615, Section 8
               (ipfixTemplateDefinitionIeLength)";
          }

          leaf ie-enterprise-number {
            type uint32;
            description
              "This parameter indicates the IANA enterprise number of
               the authority defining the Information Element
               identifier.

               If the Information Element is not enterprise-specific,
               this state parameter is zero.

               Note that this parameter corresponds to
               ipfixTemplateDefinitionIeEnterpriseNumber in the IPFIX
               MIB module.";
            reference
              "RFC 6615, Section 8
               (ipfixTemplateDefinitionIeEnterpriseNumber);
               IANA registry for Private Enterprise Numbers,
               http://www.iana.org/assignments/enterprise-numbers";
          }

          leaf is-flow-key {
            when "../../set-id = 2" {
              description
                "This parameter is available for non-Options Templates
                 (Set ID is 2).";
            }
            type empty;
            description
              "If present, this is a Flow Key field.

               Note that this corresponds to flowKey(1) being set in
               ipfixTemplateDefinitionFlags.";
            reference
              "RFC 6615, Section 8
               (ipfixTemplateDefinitionFlags)";
          }
```

```
           leaf is-scope {
             when "../../set-id = 3" {
               description
                 "This parameter is available for Options
                  Templates (Set ID is 3).";
             }
             type empty;
             description
               "If present, this is a scope field.

                Note that this corresponds to scope(0) being set in
                ipfixTemplateDefinitionFlags.";
             reference
               "RFC 6615, Section 8
                (ipfixTemplateDefinitionFlags)";
           }
         }
       }
     }

     grouping common-collector-parameters {
       description
         "Parameters of a Collecting Process that are common to all
          transport protocols.";

       choice local-address-method {
         description
           "Method to configure the local address of the collecting
            process.  Note that it is expected that other methods be
            available.  Those method can augment this choice.";

         case local-address {
           leaf-list local-address {
             type inet:host;
             description
               "List of local addresses on which the Collecting
                Process listens for IPFIX Messages.";
           }
         }
       }

       leaf local-port {
         type inet:port-number;
         description
           "If not configured, the Monitoring Device uses the default
            port number for IPFIX, which is 4739 without TLS or DTLS and
            4740 if TLS or DTLS is activated.";
       }
```

```
     }

     grouping tcp-collector-parameters {
       description
         "Parameters of a listening TCP socket at a Collecting
          Process.";

       uses common-collector-parameters;

       uses transport-layer-security-parameters;
     }

     grouping udp-collector-parameters {
       description
         "Parameters of a listening UDP socket at a Collecting
          Process.";

       uses common-collector-parameters;

       leaf template-life-time {
         type uint32;
         units seconds;
         default 1800;
         description
           "Sets the lifetime of Templates for all UDP Transport
            Sessions terminating at this UDP socket. Templates that are
            not received again within the configured lifetime become
            invalid at the Collecting Process.

            As specified in RFC 7011, the Template lifetime MUST be at
            least three times higher than the template-refresh-timeout
            parameter value configured on the corresponding Exporting
            Processes.

            Note that this parameter corresponds to
            ipfixTransportSessionTemplateRefreshTimeout in the IPFIX
            MIB module.";
         reference
           "RFC 7011, Section 10.3.7; RFC 6615, Section 8
            (ipfixTransportSessionTemplateRefreshTimeout).";
       }

       leaf options-template-life-time {
         type uint32;
         units seconds;
         default 1800;
         description
           "Sets the lifetime of Options Templates for all UDP Transport
```

```
           Sessions terminating at this UDP socket. Options Templates
           that are not received again within the configured lifetime
           become invalid at the Collecting Process.

           As specified in RFC 7011, the Options Template lifetime MUST
           be at least three times higher than the
           options-template-refresh-timeout parameter value configured
           on the corresponding Exporting Processes.

           Note that this parameter corresponds to
           ipfixTransportSessionOptionsTemplateRefreshTimeout in the
           IPFIX MIB module.";
      reference
        "RFC 7011, Section 8.4; RFC 6615, Section 8
         (ipfixTransportSessionOptionsTemplateRefreshTimeout).";
    }

    leaf template-life-packet {
      type uint32;
      units "IPFIX Messages";
      description
        "If this parameter is configured, Templates defined in a UDP
         Transport Session become invalid if they are neither
         included in a sequence of more than this number of IPFIX
         Messages nor received again within the period of time
         specified by template-life-time.

         Note that this parameter corresponds to
         ipfixTransportSessionTemplateRefreshPacket in the IPFIX
         MIB module.";
      reference
        "RFC 7011, Section 8.4; RFC 6615, Section 8
         (ipfixTransportSessionTemplateRefreshPacket).";
    }

    leaf options-template-life-packet {
      type uint32;
      units "IPFIX Messages";
      description
        "If this parameter is configured, Options Templates defined
         in a UDP Transport Session become invalid if they are
         neither included in a sequence of more than this number of
         IPFIX Messages nor received again within the period of time
         specified by options-template-life-time.

         Note that this parameter corresponds to
         ipfixTransportSessionOptionsTemplateRefreshPacket in the
         IPFIX MIB module.";
```

```
          reference
            "RFC 7011, Section 8.4; RFC 6615, Section 8
             (ipfixTransportSessionOptionsTemplateRefreshPacket).";
        }

        leaf maximum-reordering-delay {
          type uint32;
          units seconds;
          description
            "The maximum delay for the template to be received at the
             collector after the data record(s) has(have) been received.
             The collector is expected to buffer the data records till
             such a time.";
          reference
            "RFC 7011, Section 8.2";
        }

        uses transport-layer-security-parameters;
      }

      grouping sctp-collector-parameters {
        description
          "Parameters of a listening SCTP socket at a Collecting
           Process.";

        uses common-collector-parameters;

        leaf maximum-reordering-delay {
          type uint32;
          units seconds;
          description
            "The maximum delay for the template to be received at the
             collector after the data record(s) has(have) been received.
             The collector is expected to buffer the data records till
             such a time.";
          reference
            "RFC 7011, Section 8.2";
        }

        uses transport-layer-security-parameters;
      }

      grouping file-reader-state-parameters {
        description
          "State Parameters for the File Reader.";

        container file-reader-state {
          config false;
```

```
        description
          "File Reader parameters.";

        leaf bytes {
          type yang:counter64;
          units octets;
          description
            "The number of bytes read by the File Reader.

             Discontinuities in the value of this counter can occur at
             re-initialization of the management system, and at other
             times as indicated by the value of
             file-reader-discontinuity-time.";
        }

        leaf messages {
          type yang:counter64;
          units "IPFIX Messages";
          description
            "The number of IPFIX Messages read by the File Reader.

             Discontinuities in the value of this counter can occur at
             re-initialization of the management system, and at other
             times as indicated by the value of
             file-reader-discontinuity-time.";
        }

        leaf records {
          type yang:counter64;
          units "Data Records";
          description
            "The number of Data Records read by the File Reader.

             Discontinuities in the value of this counter can occur at
             re-initialization of the management system, and at other
             times as indicated by the value of
             file-reader-discontinuity-time.";
        }

        leaf templates {
          type yang:counter32;
          units "Templates";
          description
            "The number of Template Records (excluding Options Template
             Records) read by the File Reader.

             Discontinuities in the value of this counter can occur at
             re-initialization of the management system, and at other
```

```
              times as indicated by the value of
              file-reader-discontinuity-time.";
         }

         leaf options-templates {
           type yang:counter32;
           units "Options Templates";
           description
             "The number of Options Template Records read by the File
              Reader.

              Discontinuities in the value of this counter can occur at
              re-initialization of the management system, and at other
              times as indicated by the value of
              file-reader-discontinuity-time.";
         }

         leaf file-reader-discontinuity-time {
           type yang:date-and-time;
           description
             "Timestamp of the most recent occasion at which one or more
              File Reader counters suffered a discontinuity.

              In contrast to discontinuity times in the IPFIX MIB
              module, the time is absolute and not relative to
              sysUpTime.";
         }

         uses collection-template-state-parameters;
       }
     }

     grouping collecting-process-parameters {
       description
         "Parameters of a Collecting Process.";

       list tcp-collector {
         if-feature tcp-transport;
         key "name";
         description
           "List of TCP receivers (sockets) on which the Collecting
            Process receives IPFIX Messages.";

         leaf name {
           type name-type;
           description
             "An arbitrary string which uniquely identifies the TCP
              collector.";
```

```
          }

          uses tcp-collector-parameters;

          list transport-session {
            key name;
            config false;
            description
              "This list contains the currently established Transport
               Sessions terminating at the given socket.";

            leaf name {
              type name-type;
              description
                "An arbitrary string which uniquely identifies the
                 transport session.";
            }

            uses transport-session-state-parameters;
            uses collection-template-state-parameters;
          }
        }

        list udp-collector {
          if-feature udp-transport;
          key "name";
          description
            "List of UDP receivers (sockets) on which the Collecting
             Process receives IPFIX Messages.";

          leaf name {
            type name-type;
            description
              "An arbitrary string which uniquely identifies the UDP
               Collector.";
          }

          uses udp-collector-parameters;

          list transport-session {
            key name;
            config false;
            description
              "This list contains the currently established Transport
               Sessions terminating at the given socket.";

            leaf name {
              type name-type;
```

```
            description
              "An arbitrary string which uniquely identifies the
               transport session.";
          }

          uses transport-session-state-parameters;
          uses collection-template-state-parameters;
        }
      }

      list sctp-collector {
        if-feature sctp-transport;
        key "name";
        description
          "List of SCTP receivers on which the Collecting Process
           receives IPFIX Messages.";

        leaf name {
          type name-type;
          description
            "An arbitrary string which uniquely identifies the SCTP
             Collector.";
        }

        uses sctp-collector-parameters;

        list transport-session {
          key name;
          config false;
          description
            "This list contains the currently established Transport
             Sessions terminating at the given socket.";

          leaf name {
            type name-type;
            description
              "An arbitrary string which uniquely identifies the
               transport session.";
          }

          leaf sctp-association-id {
            type uint32;
            config false;
            description
              "The association ID used for the SCTP session between the
               Exporter and the Collector of the IPFIX Transport
               Session.  It is equal to the sctpAssocId entry in the
               sctpAssocTable defined in the SCTP-MIB.
```

```
                This parameter is only available if the transport
                protocol is SCTP and if an SNMP agent on the same
                Monitoring Device enables access to the corresponding
                MIB objects in the sctpAssocTable.

                Note that this parameter corresponds to
                ipfixTransportSessionSctpAssocId in the IPFIX MIB
                module.";
             reference
               "RFC 6615, Section 8
                (ipfixTransportSessionSctpAssocId);
                RFC 3871";
           }

           uses transport-session-state-parameters;
           uses collection-template-state-parameters;
         }
       }

       list file-reader {
         if-feature file-reader;
         key "name";
         description
           "List of File Readers from which the Collecting Process reads
            the IPFIX Messages.";

         leaf name {
           type name-type;
           description
             "An arbitrary string which uniquely identifies the File
              Reader.";
         }

         leaf file {
           type inet:uri;
           mandatory true;
           description
             "URI specifying the location of the file.";
         }

         uses file-reader-state-parameters;
       }
     }

     grouping export-template-state-parameters {
       description
         "State parameters of a (Options) Template used by an Exporting
          Process in a specific Transport Session or by a File Writer.
```

```
          Parameter names and semantics correspond to the managed
          objects in IPFIX-MIB.";
       reference
         "RFC 7011; RFC 6615, Section 8 (ipfixTemplateEntry,
          ipfixTemplateDefinitionEntry, ipfixTemplateStatsEntry)";

       list template {
         key "name";
         description
           "This list contains the Templates and Options Templates that
            are transmitted by the Exporting Process or written by the
            File Writer.

            Withdrawn or invalidated (Options) Templates MUST be removed
            from this list.";

         leaf name {
           type name-type;
           description
             "An arbitrary string which uniquely identifies the
              template.";
         }

         leaf observation-domain-id {
           type uint32;
           description
             "The ID of the Observation Domain for which this Template
              is defined.

              Note that this parameter corresponds to
              ipfixTemplateObservationDomainId in the IPFIX MIB
              module.";
           reference
             "RFC 6615, Section 8
              (ipfixTemplateObservationDomainId).";
         }

         leaf template-id {
           type uint16 {
             range "256..65535";
           }
           description
             "This number indicates the Template ID in the IPFIX
              message.

              Note that this parameter corresponds to ipfixTemplateId in
              the IPFIX MIB module.";
           reference
```

```
              "RFC 6615, Section 8 (ipfixTemplateId).";
          }

          leaf set-id {
            type uint16 {
              range "2..3 | 256..65535";
            }
            description
              "This number indicates the Set ID of the Template.
               A value of 2 is reserved for Template Sets.  A value of 3
               is reserved for Options Template Sets.  Values from 4 to
               255 are reserved for future use.  Values 256 and above
               are used for Data Sets.  The Set ID values of 0 and 1 are
               not used for historical reasons.

               Note that this parameter corresponds to ipfixTemplateSetId
               in the IPFIX MIB module.";
            reference
              "RFC 7011, Section 3.3.2;
               RFC 6615, Section 8 (ipfixTemplateSetId)";
          }

          leaf access-time {
            type yang:date-and-time;
            description
              "This parameter contains the time when this (Options)
               Template was last sent to the Collector(s) or written to
               the file.

               Note that this parameter corresponds to
               ipfixTemplateAccessTime in the IPFIX MIB module.";
            reference
              "RFC 6615, Section 8 (
               ipfixTemplateAccessTime).";
          }

          leaf template-data-records {
            type yang:counter64;
            description
              "The number of transmitted Data Records defined by this
               (Options) Template.

               Discontinuities in the value of this counter can occur at
               re-initialization of the management system, and at other
               times as indicated by the value of
               template-discontinuity-time.

               Note that this parameter corresponds to
```

```
              ipfixTemplateDataRecords in the IPFIX MIB module.";
            reference
              "RFC 6615, Section 8 (ipfixTemplateDataRecords).";
          }

          leaf template-discontinuity-time {
            type yang:date-and-time;
            description
              "Timestamp of the most recent occasion at which the counter
               template-data-records suffered a discontinuity.

               Note that this parameter functionally
               corresponds to ipfixTemplateDiscontinuityTime in the IPFIX
               MIB module. In contrast to
               ipfixTemplateDiscontinuityTime, the time is absolute and
               not relative to sysUpTime.";
            reference
              "RFC 6615, Section 8
               (ipfixTemplateDiscontinuityTime).";
          }

          list field {
            key "name";
            description
              "This list contains the (Options) Template fields of which
               the (Options) Template is defined.

               The order of the list corresponds to the order
               of the fields in the (Option) Template Record.";

            leaf name {
              type name-type;
              description
                "An arbitrary string which uniquely identifies the
                 template field.";
            }

            leaf ie-id {
              type ie-id-type;
              description
                "This parameter indicates the Information Element
                 identifier of the field.

                 Note that this parameter corresponds to
                 ipfixTemplateDefinitionIeId in the IPFIX MIB module.";
              reference
                "RFC 7011; RFC 6615, Section 8
                 (ipfixTemplateDefinitionIeId).";
```

```
           }

           leaf ie-length {
             type uint16;
             units "octets";
             description
               "This parameter indicates the length of the Information
                Element of the field.

                Note that this parameter corresponds to
                ipfixTemplateDefinitionIeLength in the IPFIX MIB
                module.";
             reference
               "RFC 7011; RFC 6615, Section 8
                (ipfixTemplateDefinitionIeLength).";
           }

           leaf ie-enterprise-number {
             type uint32;
             description
               "This parameter indicates the IANA enterprise number of
                the authority defining the Information Element
                identifier.

                If the Information Element is not enterprise-specific,
                this state parameter is zero.

                Note that this parameter corresponds to
                ipfixTemplateDefinitionIeEnterpriseNumber in the IPFIX
                MIB module.";
             reference
               "RFC 6615, Section 8
                (ipfixTemplateDefinitionIeEnterpriseNumber);
                IANA registry for Private Enterprise Numbers,
                http://www.iana.org/assignments/enterprise-numbers.";
           }

           leaf is-flow-key {
             when "../../set-id = 2" {
             description
               "This parameter is available for non-Options Templates
                (Set ID is 2).";
             }
             type empty;
             description
               "If present, this is a Flow Key field.

                Note that this corresponds to flowKey(1) being set in
```

```
                ipfixTemplateDefinitionFlags.";
            reference
              "RFC 6615, Section 8
               (ipfixTemplateDefinitionFlags).";
          }

          leaf is-scope {
            when "../../set-id = 3" {
            description
              "This parameter is available for Options Templates
               (Set ID is 3).";
            }
            type empty;
            description
              "If present, this is a scope field.

               Note that this corresponds to scope(0) being set in
               ipfixTemplateDefinitionFlags.";
            reference
              "RFC 6615, Section 8
               (ipfixTemplateDefinitionFlags).";
          }
        }
      }
    }

    grouping common-exporter-parameters {
      description
        "Parameters of an export destination that are common to all
         transport protocols.";

      leaf ipfix-version {
        type uint16;
        default '10';
        description
          "IPFIX version number.";
        reference
          "RFC 7011.";
      }

      container source {
        description
          "Configuration corresponding to how exporter's source IP
           address is specified.";

        choice source-method {
          description
            "Method to configure the source address of the exporter
```

```
              or the interface to be used by the exporter.

              Note that it is expected that other methods be available.
              Those methods can augment this choice.";

           case interface-ref {
             leaf interface-ref {
               type if:interface-ref;
               description
                 "The interface to be used by the Exporting Process.";
             }
           }

           case if-index {
             if-feature if-mib;
             leaf if-index {
               type uint32;
               description
                 "Index of an interface as stored in the ifTable
                  of IF-MIB.";
               reference
                 "RFC 2863.";
             }
           }

           case if-name {
             if-feature if-mib;
             leaf if-name {
               type string;
               description
                 "Name of an interface as stored in the ifTable
                  of IF-MIB.";
               reference
                 "RFC 2863.";
             }
           }
         }
       }

       container destination {
         description
           "Configuration corresponding to how exporter's destination IP
            address is specified.";
       }

       leaf destination-port {
         type inet:port-number;
         description
```

```
           "If not configured by the user, the Monitoring Device uses
            the default port number for IPFIX, which is 4739 without TLS
            or DTLS and 4740 if TLS or DTLS is activated.";
       }

       leaf send-buffer-size {
         type uint32;
         units "bytes";
         description
           "Size of the socket send buffer.

            If not configured by the user, this parameter is set by
            the Monitoring Device.";
       }

       leaf rate-limit {
         type uint32;
         units "bytes per second";
         description
           "Maximum number of bytes per second the Exporting Process may
            export to the given destination.  The number of bytes is
            calculated from the lengths of the IPFIX Messages exported.
            If not configured, no rate limiting is performed.";
         reference
           "RFC 5476, Section 6.3.";
       }
     }

     grouping tcp-exporter-parameters {
       description
         "Parameters of a TCP export destination.";

       uses common-exporter-parameters {
         augment "source/source-method" {
           description
             "Augment the source method to add the source IP address or
              hostname.";

           case source-address {
             leaf source-address {
               type inet:host;
               description
                 "The source IP address or hostname used by the
                  Exporting Process.";
             }
           }
         }
```

```
          augment "destination" {
            description
              "Augment the destination method to add the destination
               IP address or hostname.";

            choice destination-method {
              mandatory true;
              description
                "Method to configuring the destination address of the
                 Collection Process to which IPFIX Messages are sent.

                 Note it is expected that if other methods are available
                 that they would augment from this statement.";

              case destination-address {
                leaf destination-address {
                  type inet:host;
                  description
                    "The destination IP address or hostname of the
                     Collecting Process to which IPFIX Messages are sent.
                     A hostname may resolve to one or more IP
                     addresses.";
                }
              }
            }
          }
        }

        leaf connection-timeout {
          type uint32;
          units seconds;
          description
            "Time after which the exporting process deems the TCP
             connection to have failed.";
          reference
            "RFC 7011, Sections 10.4.4 and 10.4.5.";
        }

        leaf retry-schedule {
          type uint32 {
          range "60..max";
          }
          units seconds;
          description
            "Time after which the exporting process retries the TCP
             connection to a collector.";
          reference
            "RFC 7011, Section 10.4.4.";
```

```
       }

       uses transport-layer-security-parameters;
     }

     grouping udp-exporter-parameters {
       description
         "Parameters of a UDP export destination.";

       uses common-exporter-parameters {
         augment "source/source-method" {
           description
             "Augment the source method to add the source IP address or
              hostname.";

           case source-address {
             leaf source-address {
               type inet:host;
               description
                 "The source IP address or hostname used by the
                  Exporting Process.";
             }
           }
         }

         augment "destination" {
           description
             "Augment the destination method to add the destination
              IP address or hostname.";

           choice destination-method {
             mandatory true;
             description
               "Method to configuring the destination address of the
                Collection Process to which IPFIX Messages are sent.

                Note it is expected that if other methods are available
                that they would augment from this statement.";

             case destination-address {
               leaf destination-address {
                 type inet:host;
                 description
                   "The destination IP address or hostname of the
                    Collecting Process to which IPFIX Messages are sent.
                    A hostname may resolve to one or more IP
                    addresses.";
               }
```

```
             }
           }
         }
       }

       leaf maximum-packet-size {
         type uint16;
         units octets;
         description
           "This parameter specifies the maximum size of IP packets sent
            to the Collector.  If set to zero, the Exporting Device MUST
            derive the maximum packet size from path MTU discovery
            mechanisms.

            If not configured by the user, this parameter is set by
            the Monitoring Device.";
       }

       leaf template-refresh-timeout {
         type uint32;
         units seconds;
         default 600;
         description
           "Sets time after which Templates are resent in the UDP
            Transport Session.

            Note that the configured lifetime MUST be adapted to the
            template-life-time parameter value at the receiving
            Collecting Process.

            Note that this parameter corresponds to
            ipfixTransportSessionTemplateRefreshTimeout in the IPFIX
            MIB module.";
         reference
           "RFC 7011, Section 8.4; RFC 6615, Section 8
            (ipfixTransportSessionTemplateRefreshTimeout).";
       }

       leaf options-template-refresh-timeout {
         type uint32;
         units seconds;
         default 600;
         description
           "Sets time after which Options Templates are resent in the
            UDP Transport Session.

            Note that the configured lifetime MUST be adapted to the
            options-template-life-time parameter value at the receiving
```

```
            Collecting Process.

            Note that this parameter corresponds to
            ipfixTransportSessionOptionsTemplateRefreshTimeout in the
            IPFIX MIB module.";
         reference
           "RFC 7011, Section 8.4; RFC 6615, Section 8
            (ipfixTransportSessionOptionsTemplateRefreshTimeout).";
       }

       leaf template-refresh-packet {
         type uint32;
         units "IPFIX Messages";
         description
           "Sets number of IPFIX Messages after which Templates are
            resent in the UDP Transport Session.

            Note that this parameter corresponds to
            ipfixTransportSessionTemplateRefreshPacket in the IPFIX
            MIB module.

            If omitted, Templates are only resent after timeout.";
         reference
           "RFC 7011, Section 8.4; RFC 6615, Section 8
            (ipfixTransportSessionTemplateRefreshPacket).";
       }

       leaf options-template-refresh-packet {
         type uint32;
         units "IPFIX Messages";
         description
           "Sets number of IPFIX Messages after which Options Templates
            are resent in the UDP Transport Session protocol.

            Note that this parameter corresponds to
            ipfixTransportSessionOptionsTemplateRefreshPacket in the
            IPFIX MIB module.

            If omitted, Templates are only resent after timeout.";
         reference
           "RFC 7011, Section 8.4; RFC 6615, Section 8
            (ipfixTransportSessionOptionsTemplateRefreshPacket).";
       }

       uses transport-layer-security-parameters;
     }

     grouping sctp-exporter-parameters {
```

```
       description
         "Parameters of a SCTP export destination.";

       uses common-exporter-parameters {
         augment "source/source-method" {
           description
             "Augment the source method to add the source IP address or
              hostname.";

           case source-address {
             leaf-list source-address {
               type inet:host;
               description
                 "The source IP address(es) or hostname(s) used by the
                  Exporting Process.";
             }
           }
         }

         augment "destination" {
           description
             "Augment the destination method to add the destination
              IP address or hostname.";

           choice destination-method {
             mandatory true;
             description
               "Method to configuring the destination address of the
                Collection Process to which IPFIX Messages are sent.

                Note it is expected that if other methods are available
                that they would augment from this statement.";

             case destination-address {
               leaf-list destination-address {
                 type inet:host;
                 description
                   "List of destination IP addresses or hostnames.
                    A hostname may resolve to one or more IP addresses.

                    The user must ensure that all configured IP
                    addresses belong to the same Collecting Process.

                    The SCTP Exporting Processs tries to establish an
                    SCTP association to any of the configured
                    destination IP addresses.";
               }
             }
```

```
          }
        }
      }

      leaf timed-reliability {
        type uint32;
        units milliseconds;
        default 0;
        description
          "Lifetime in milliseconds until an IPFIX Message containing
           Data Sets only is 'abandoned' due to the timed reliability
           mechanism of PR-SCTP.

           If this parameter is set to zero, reliable SCTP transport is
           used for all Data Records.

           Regardless of the value of this parameter, the Exporting
           Process MAY use reliable SCTP transport for Data Sets
           associated with Options Templates.";
        reference
          "RFC 3758; RFC 4960.";
      }

      leaf association-timeout {
        type uint32;
        units seconds;
        description
          "Time after which the exporting process deems the SCTP
           association to have failed.";
        reference
          "RFC 7011, Sections 10.2.4 and 10.2.5.";
      }

      uses transport-layer-security-parameters;
    }

    grouping file-writer-state-parameters {
      description
        "State Parameters for the File Writer.";

      container file-writer-state {
        config false;
        description
          "File Writer parameters.";

        leaf bytes {
          type yang:counter64;
          units octets;
```

```
          description
            "The number of bytes written by the File Writer.

             Discontinuities in the value of this counter can occur at
             re-initialization of the management system, and at other
             times as indicated by the value of
             file-writer-discontinuity-time.";
        }

        leaf messages {
          type yang:counter64;
          units "IPFIX Messages";
          description
            "The number of IPFIX Messages written by the File Writer.

             Discontinuities in the value of this counter can occur at
             re-initialization of the management system, and at other
             times as indicated by the value of
             file-writer-discontinuity-time.";
        }

        leaf discarded-messages {
          type yang:counter64;
          units "IPFIX Messages";
          description
            "The number of IPFIX Messages that could not be written by
             the File Writer due to internal buffer overflows, limited
             storage capacity, etc.

             Discontinuities in the value of this counter can occur at
             re-initialization of the management system, and at other
             times as indicated by the value of
             file-writer-discontinuity-time.";
        }

        leaf records {
          type yang:counter64;
          units "Data Records";
          description
            "The number of Data Records written by the File Writer.

             Discontinuities in the value of this counter can occur at
             re-initialization of the management system, and at other
             times as indicated by the value of
             file-writer-discontinuity-time.";
        }

        leaf templates {
```

```
          type yang:counter32;
          units "Templates";
          description
            "The number of Template Records (excluding Options Template
             Records) written by the File Writer.

             Discontinuities in the value of this counter can occur at
             re-initialization of the management system, and at other
             times as indicated by the value of
             file-writer-discontinuity-time.";
        }

        leaf options-templates {
          type yang:counter32;
          units "Options Templates";
          description
            "The number of Options Template Records written by the File
             Writer.

             Discontinuities in the value of this counter can occur at
             re-initialization of the management system, and at other
             times as indicated by the value of
             file-writer-discontinuity-time.";
        }

        leaf file-writer-discontinuity-time {
          type yang:date-and-time;
          description
            "Timestamp of the most recent occasion at which one or more
             File Writer counters suffered a discontinuity.

             In contrast to discontinuity times in the IPFIX MIB
             module, the time is absolute and not relative to
             sysUpTime.";
        }

        uses export-template-state-parameters;
      }
    }

    grouping exporting-process-parameters {
      description
        "Parameters of an Exporting Process.";

      leaf export-mode {
        type identityref {
          base export-mode;
        }
```

```
          default 'parallel';
          description
            "This parameter determines to which configured destination(s)
             the incoming Data Records are exported.";
        }

        list destination {
          key "name";
          min-elements 1;
          description
            "List of export destinations.";

          leaf name {
            type name-type;
            description
              "An arbitrary string which uniquely identifies the export
               destination.";
          }

          choice destination-parameters {
            mandatory true;
            description
              "Destination configuration.";

            case tcp-exporter {
              container tcp-exporter {
                if-feature tcp-transport;
                description
                  "TCP parameters.";

                uses tcp-exporter-parameters;

                container transport-session {
                  config false;
                  description
                    "Transport session state data.";

                  uses transport-session-state-parameters;
                  uses export-template-state-parameters;
                }
              }
            }

            case udp-exporter {
              container udp-exporter {
                if-feature udp-transport;
                description
                  "UDP parameters.";
```

```
              uses udp-exporter-parameters;

              container transport-session {
                config false;
                description
                  "Transport session state data.";

                uses transport-session-state-parameters;
                uses export-template-state-parameters;
              }
            }
          }

          case sctp-exporter {
            container sctp-exporter {
              if-feature sctp-transport;
              description
                "SCTP parameters.";

              uses sctp-exporter-parameters;

              container transport-session {
                config false;
                description
                  "Transport session state data.";

                leaf sctp-association-id {
                  type uint32;
                  description
                    "The association ID used for the SCTP session
                     between the Exporter and the Collector of the
                     IPFIX Transport Session.  It is equal to the
                     sctpAssocId entry in the sctpAssocTable defined in
                     the SCTP-MIB.

                     This parameter is only available if the transport
                     protocol is SCTP and if an SNMP agent on the same
                     Monitoring Device enables access to the
                     corresponding MIB objects in the sctpAssocTable.

                     Note that this parameter corresponds to
                     ipfixTransportSessionSctpAssocId in the IPFIX MIB
                     module.";
                  reference
                    "RFC 6615, Section 8
                     (ipfixTransportSessionSctpAssocId);
                     RFC 3871";
                }
```

```
                 uses transport-session-state-parameters;
                 uses export-template-state-parameters;
               }
             }
           }

           case file-writer {
             container file-writer {
               if-feature file-writer;
               description
                 "File Writer parameters.";

               leaf ipfix-version {
                 type uint16;
                 default 10;
                 description
                   "IPFIX version number.";
                 reference
                   "RFC 7011.";
               }

               leaf file {
                 type inet:uri;
                 mandatory true;
                 description
                   "URI specifying the location of the file.";
               }

               uses file-writer-state-parameters;
             }
           }
         }
       }

       list options {
         key "name";
         description
           "List of options reported by the Exporting Process.";

         leaf name {
           type name-type;
           description
             "An arbitrary string which uniquely identifies the
              option.";
         }
         uses options-parameters;
       }
     }
```

```
     grouping options-parameters {
       description
         "Parameters specifying the data export using an Options
          Template.";

       leaf options-type {
         type identityref {
           base options-type;
         }
         mandatory true;
         description
           "Type of the exported options data.";
       }

       leaf options-timeout {
         type uint32;
         units "milliseconds";
         description
           "Time interval for periodic export of the options data. If
            set to zero, the export is triggered when the options data
            has changed.

            If not configured by the user, this parameter is set by the
            Monitoring Device.";
       }
     }

     container ipfix {
       description
         "IPFIX Exporter and/or Collector data nodes.";

       list collecting-process {
         if-feature collector;
         key "name";
         description
           "Collecting Process of the Monitoring Device.";

         leaf name {
           type name-type;
           description
             "An arbitrary string which uniquely identifies the
              Collecting Process.";
         }

         uses collecting-process-parameters;

         leaf-list exporting-process {
           if-feature exporter;
```

```
          type leafref {
            path "/ietf-ipfix:ipfix"
               + "/ietf-ipfix:exporting-process"
               + "/ietf-ipfix:name";
          }
          description
            "Export of received records without any modifications.
             Records are processed by all Exporting Processes in the
             list.";
        }
      }

      list exporting-process {
        if-feature exporter;
        key "name";
        description
          "List of Exporting Processes of the IPFIX Monitoring Device
           for which configuration will be applied.";

        leaf name {
          type name-type;
          description
            "An arbitrary string which uniquely identifies the
             Exporting Process.";
        }

        leaf enabled {
          type boolean;
          default "true";
          description
            "If true, this Exporting Process is enabled for
             exporting.";
        }

        uses exporting-process-parameters;

        leaf exporting-process-id {
          type uint32;
          config false;
          description
            "The identifier of the Exporting Process.  This parameter
             corresponds to the Information Element exportingProcessId.
             Its occurrence helps to associate Exporting Process
             parameters with Exporing Process statistics exported by
             the Monitoring Device using the Exporting Process
             Reliability Statistics Template as defined by the IPFIX
             protocol specification.";
          reference
```

```
             "RFC 7011, Section 4.3; IANA registry for IPFIX
              Entities, http://www.iana.org/assignments/ipfix.";
        }
      }
    }
  }

  <CODE ENDS>
```

6.2. ietf-ipfix-packet-sampling

6.2.1. ietf-ipfix-packet-sampling Module Structure

This document defines the YANG module "ietf-ipfix-packet-sampling", which has the following structure:

```
   module: ietf-ipfix-packet-sampling
     augment /ipfix:ipfix:
       +--rw psamp
          +--rw observation-point* [name]
          |  +--rw name                     ipfix:name-type
          |  +--rw observation-domain-id    uint32
          |  +--rw interface-ref*           if:interface-ref
          |  +--rw if-name*                 if-name-type {if-mib}?
          |  +--rw if-index*                uint32 {if-mib}?
          |  +--rw hardware-ref*            hardware-ref
          |  +--rw ent-physical-name*       string {entity-mib}?
          |  +--rw ent-physical-index*      uint32 {entity-mib}?
          |  +--rw direction?               direction
          |  +--rw selection-process*
          |  |       -> /ipfix:ipfix/psamp/selection-process/name
          |  +--ro observation-point-id?    uint32
          +--rw selection-process* [name]
          |  +--rw name                  ipfix:name-type
          |  +--rw selector* [name]
          |  |     ...
          |  +--rw cache?
          |  |       -> /ipfix:ipfix/psamp/cache/name
          |  +--ro selection-sequence* []
          |        ...
          +--rw cache* [name]
             +--rw name                        ipfix:name-type
             +--rw enabled?                    boolean
             +--rw (cache-type)
             |     ...
             +--rw exporting-process*
             |       -> /ipfix:ipfix/exporting-process/name
             |       {ipfix:exporter}?
             +--ro metering-process-id?        uint32
             +--ro data-records?               yang:counter64
             +--ro cache-discontinuity-time?   yang:date-and-time
```

6.2.2. ietf-ipfix-packet-sampling YANG module

This YANG Module imports typedefs from [RFC6991].

```
   <CODE BEGINS> file "ietf-ipfix-packet-sampling@2018-10-22.yang"

   module ietf-ipfix-packet-sampling {
     yang-version 1.1;

     namespace "urn:ietf:params:xml:ns:yang:ietf-ipfix-packet-sampling";

     prefix ips;
```

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

     import ietf-interfaces {
       prefix if;
       reference
         "RFC 8343: A YANG Model for Interface Management";
     }

     import ietf-hardware {
       prefix hw;
       reference
         "RFC 8348: A YANG Data Model for Hardware Management";
     }

     import ietf-ipfix {
       prefix ipfix;
       reference
         "RFC XXXX: YANG Data Models for the IP Flow Information Export
          (IPFIX) Protocol, Packet Sampling (PSAMP) Protocol, and Bulk
          Data Export";
     }

     organization
       "IETF";

     contact
       "Web:      TBD
        List:     TBD

        Editor:   Joey Boyd
                  <mailto:joey.boyd@adtran.com>

        Editor:   Marta Seda
                  <mailto:marta.seda@calix.com>";

     // RFC Ed.: replace XXXX with actual RFC numbers and
     // remove this note.

     description
       "This module contains a collection of YANG definitions for the
        management Packet Sampling (PSAMP) over IPFIX.

        This data model is designed for the Network Management Datastore
        Architecture defined in RFC 8342.
```

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

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

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

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

     revision 2020-03-05 {
       description
         "Initial revision.";
       reference
         "RFC XXXX: YANG Data Models for the IP Flow Information Export
                    (IPFIX) Protocol, Packet Sampling (PSAMP) Protocol,
                    and Bulk Data Export";
     }

     feature if-mib {
       description
         "This feature indicates that the device implements the
          IF-MIB.";
       reference
         "RFC 2863: The Interfaces Group MIB";
     }

     feature entity-mib {
       description
         "This feature indicates that the device implements the
          ENTITY-MIB.";
       reference
         "RFC 6933: Entity MIB (Version 4)";
     }

     feature psamp-samp-count-based {
       description
         "If supported, the Monitoring Device supports count-based
```

```
          sampling. The Selector method sampCountBased can be
          configured.";
      reference
        "RFC 5475, Section 5.1";
    }

    feature psamp-samp-time-based {
      description
        "If supported, the Monitoring Device supports time-based
         sampling. The Selector method sampTimeBased can be
         configured.";
      reference
        "RFC 5475, Section 5.1";
    }

    feature psamp-samp-rand-out-of-n {
      description
        "If supported, the Monitoring Device supports random n-out-of-N
         sampling. The Selector method sampRandOutOfN can be
         configured.";
      reference
        "RFC 5475, Section 5.2.1";
    }

    feature psamp-samp-uni-prob {
      description
        "If supported, the Monitoring Device supports uniform
         probabilistic sampling. The Selector method sampUniProb can be
         configured.";
      reference
        "RFC 5475, Section 5.2.2";
    }

    feature psamp-filter-match {
      description
        "If supported, the Monitoring Device supports property match
         filtering. The Selector method filterMatch can be
         configured.";
      reference
        "RFC 5475, Section 6.1";
    }

    feature psamp-filter-hash {
      description
        "If supported, the Monitoring Device supports hash-based
         filtering. The Selector method filterHash can be configured.";
      reference
        "RFC 5475, Section 6.2";
```

```
     }

     feature immediate-cache {
       description
         "If supported, the Monitoring Device supports
          Caches generating PSAMP Packet Reports by configuration with
          immediateCache.";
     }

     feature timeout-cache {
       description
         "If supported, the Monitoring Device supports
          Caches generating IPFIX Flow Records by configuration with
          timeoutCache.";
     }

     feature natural-cache {
       description
         "If supported, the Monitoring Device supports
          Caches generating IPFIX Flow Records by configuration with
          naturalCache.";
     }

     feature permanent-cache {
       description
         "If supported, the Monitoring Device supports
          Caches generating IPFIX Flow Records by configuration with
          permanentCache.";
     }

     identity hash-function {
       description
         "Base identity for all hash functions used for
          hash-based packet Filtering.";
     }

     identity bob {
       base hash-function;
       description
         "BOB hash function.";
       reference
         "RFC 5475, Section 6.2.4.1";
     }

     identity ipsx {
       base hash-function;
       description
         "IPSX hash function.";
```

```
       reference
         "RFC 5475, Section 6.2.4.1";
     }

     identity crc {
       base hash-function;
       description
         "CRC hash function.";
       reference
         "RFC 5475, Section 6.2.4.1";
     }

     typedef hardware-ref {
       type leafref {
         path "/hw:hardware/hw:component/hw:name";
       }
       description
         "This type is used to reference hardware components.";
       reference
         "RFC 8348";
     }

     typedef if-name-type {
       type string {
         length "1..255";
       }
       description
         "This corresponds to the DisplayString textual
          convention of SNMPv2-TC, which is used for ifName in the IF
          MIB module.";
       reference
         "RFC 2863 (ifName)";
     }

     typedef direction {
       type enumeration {
         enum "ingress" {
           value 0;
           description
             "This value is used for monitoring incoming packets.";
         }
         enum "egress" {
           value 1;
           description
             "This value is used for monitoring outgoing packets.";
         }
         enum "both" {
           value 2;
```

```
            description
              "This value is used for monitoring incoming and outgoing
               packets.";
          }
        }
        description
          "Direction of packets going through an interface.";
      }

      grouping observation-point-parameters {
        description
          "Interface as input to Observation Point.";

        leaf observation-domain-id {
          type uint32;
          mandatory true;
          description
            "The Observation Domain ID associates the Observation Point
             to an Observation Domain. Observation Points with identical
             Observation Domain IDs belong to the same Observation
             Domain.

             Note that this parameter corresponds to
             ipfixObservationPointObservationDomainId in the IPFIX MIB
             module.";
          reference
            "RFC 7011; RFC 6615, Section 8
             (ipfixObservationPointObservationDomainId)";
        }

        leaf-list interface-ref {
          type if:interface-ref;
          description
            "List of interfaces of the Monitoring Device. The
             Observation Point observes packets at the specified
             interfaces.";
        }

        leaf-list if-name {
          if-feature if-mib;
          type if-name-type;
          description
            "List of names identifying interfaces of the Monitoring
             Device.  The Observation Point observes packets at the
             specified interfaces.";
        }

        leaf-list if-index {
```

```
         if-feature if-mib;
         type uint32;
         description
           "List of if-index values pointing to entries in the ifTable
            of the IF-MIB module maintained by the Monitoring
            Device.  The Observation Point observes packets at the
            specified interfaces.

            This parameter SHOULD only be used if an SNMP agent enables
            access to the ifTable.

            Note that this parameter corresponds to
            ipfixObservationPointPhysicalInterface in the IPFIX MIB
            module.";
         reference
           "RFC 2863; RFC 6615, Section 8
            (ipfixObservationPointPhysicalInterface)";
       }

       leaf-list hardware-ref {
         type hardware-ref;
         description
           "List of hardware components of the Monitoring Device.
            The Observation Points observes packets at the specified
            hardware components.";
         reference
           "RFC 8348";
     }

       leaf-list ent-physical-name {
         if-feature entity-mib;
         type string;
         description
           "List of names identifying physical entities of the
            Monitoring Device. The Observation Point observes packets
            at the specified entities.";
       }

       leaf-list ent-physical-index {
         if-feature entity-mib;
         type uint32;
         description
           "List of ent-physical-index values pointing to entries in the
            entPhysicalTable of the ENTITY-MIB module maintained by the
            Monitoring Device.  The Observation Point observes packets
            at the specified entities.

            This parameter SHOULD only be used if an SNMP agent enables
```

```
            access to the entPhysicalTable.

            Note that this parameter corresponds to
            ipfixObservationPointPhysicalEntity in the IPFIX MIB
            module.";
         reference
           "RFC 33; RFC 6615, Section 8
            (ipfixObservationPointPhysicalInterface)";
       }

       leaf direction {
         type direction;
         default "both";
         description
           "Direction of packets. If not applicable (e.g., in the case
            of a sniffing interface in promiscuous mode), this
            parameter is ignored.";
       }
     }

     grouping samp-count-based-parameters {
       description
         "Configuration parameters of a Selector applying systematic
          count-based packet Sampling to the packet stream.";
       reference
         "RFC 5475, Section 5.1; RFC 5476, Section 6.5.2.1.";

       leaf packet-interval {
         type uint32;
         units "packets";
         mandatory true;
         description
           "The number of packets that are consecutively sampled between
            gaps of length packetSpace.

            This parameter corresponds to the Information Element
            samplingPacketInterval and to psampSampCountBasedInterval
            in the PSAMP MIB module.";
         reference
           "RFC 5477, Section 8.2.2; RFC 6727, Section 6
            (psampSampCountBasedInterval)";
       }

       leaf packet-space {
         type uint32;
         units "packets";
         mandatory true;
         description
```

```
            "The number of unsampled packets between two Sampling
             intervals.

             This parameter corresponds to the Information Element
             samplingPacketSpace and to psampSampCountBasedSpace
             in the PSAMP MIB module.";
          reference
            "RFC 5477, Section 8.2.3; RFC 6727, Section 6
             (psampSampCountBasedSpace)";
        }
      }

      grouping samp-time-based-parameters {
        description
          "Configuration parameters of a Selector applying systematic
           time-based packet Sampling to the packet stream.";
        reference
          "RFC 5475, Section 5.1; RFC 5476, Section 6.5.2.2";

        leaf time-interval {
          type uint32;
          units "microseconds";
          mandatory true;
          description
            "The time interval in microseconds during which all arriving
             packets are sampled between gaps of length timeSpace.

             This parameter corresponds to the Information Element
             samplingTimeInterval and to psampSampTimeBasedInterval
             in the PSAMP MIB module.";
          reference
            "RFC 5477, Section 8.2.4; RFC 6727, Section 6
             (psampSampTimeBasedInterval)";
        }

        leaf time-space {
          type uint32;
          units "microseconds";
          mandatory true;
          description
            "The time interval in microseconds during which no packets
             are sampled between two Sampling intervals specified by
             timeInterval.

             This parameter corresponds to the Information Element
             samplingTimeInterval and to psampSampTimeBasedSpace
             in the PSAMP MIB module.";
          reference
```

```
           "RFC 5477, Section 8.2.5; RFC 6727, Section 6
            (psampSampTimeBasedSpace)";
       }
     }

     grouping samp-rand-out-of-n-parameters {
       description
         "Configuration parameters of a Selector applying n-out-of-N
          packet Sampling to the packet stream.";
       reference
         "RFC 5475, Section 5.2.1; RFC 5476, Section 6.5.2.3.";

       leaf size {
         type uint32;
         units "packets";
         mandatory true;
         description
           "The number of elements taken from the parent population.

            This parameter corresponds to the Information Element
            samplingSize and to psampSampRandOutOfNSize in the PSAMP
            MIB module.";
         reference
           "RFC 5477, Section 8.2.6; RFC 6727, Section 6
            (psampSampRandOutOfNSize)";
       }

       leaf population {
         type uint32;
         units "packets";
         mandatory true;
         description
           "The number of elements in the parent population.

            This parameter corresponds to the Information Element
            samplingPopulation and to psampSampRandOutOfNPopulation
            in the PSAMP MIB module.";
         reference
           "RFC 5477, Section 8.2.7; RFC 6727, Section 6
            (psampSampRandOutOfNPopulation)";
       }
     }

     grouping samp-uni-prob-parameters {
       description
         "Configuration parameters of a Selector applying uniform
          probabilistic packet Sampling (with equal probability per
          packet) to the packet stream.";
```

```
       reference
         "RFC 5475, Section 5.2.2.1;
          RFC 5476, Section 6.5.2.4";

       leaf probability {
         type decimal64 {
           fraction-digits 18;
           range "0..1";
         }
         mandatory true;
         description
           "Probability that a packet is sampled, expressed as a value
            between 0 and 1.  The probability is equal for every
            packet.

            This parameter corresponds to the Information Element
            samplingProbability and to psampSampUniProbProbability
            in the PSAMP MIB module.";
         reference
           "RFC 5477, Section 8.2.8; RFC 6727, Section 6
            (psampSampUniProbProbability)";
       }
     }

     grouping filter-match-parameters {
       description
         "Configuration parameters of a Selector applying property match
          Filtering to the packet stream.

          The field to be matched is specified as an Information
          Element.";
       reference
         "RFC 5475, Section 6.1; RFC 5476, Section 6.5.2.5";

       choice information-element {
         mandatory true;
         description
           "The Information Element field to be matched.";

         case ie-name {
           leaf ie-name {
             type ipfix:ie-name-type;
             description
               "Name of the Information Element.";
           }
         }

         case ie-id {
```

```
          leaf ie-id {
            type ipfix:ie-id-type;
            description
              "ID of the Information Element.";
          }
        }
      }

      leaf ie-enterprise-number {
        type uint32;
        default '0';
        description
          "If this parameter is zero, the Information Element is
           registered in the IANA registry of IPFIX Information
           Elements.

           If this parameter is configured with a non-zero private
           enterprise number, the Information Element is
           enterprise-specific.";
        reference
          "IANA registry for Private Enterprise Numbers,
           http://www.iana.org/assignments/enterprise-numbers;
           IANA registry for IPFIX Entities,
           http://www.iana.org/assignments/ipfix.";
      }

      leaf value {
        type string;
        mandatory true;
        description
          "Matching value of the Information Element";
      }
    }

    grouping filter-hash-parameters {
      description
        "Configuration parameters of a Selector applying hash-based
         Filtering to the packet stream.";
      reference
        "RFC 5475, Section 6.2; RFC 5476, Section 6.5.2.6";

      leaf hash-function {
        type identityref {
          base hash-function;
        }
        default 'bob';
        description
          "Hash function to be applied.  According to RFC 5475,
```

```
            Section 6.2.4.1, 'BOB' must be used in order to be compliant
            with PSAMP.

            This parameter functionally corresponds to
            psampFiltHashFunction in the PSAMP MIB module.";
         reference
           "RFC 6727, Section 6 (psampFiltHashFunction)";
       }

       leaf initializer-value {
         type uint64;
         description
           "Initializer value to the hash function.
            If not configured by the user, the Monitoring Device
            arbitrarily chooses an initializer value.

            This parameter corresponds to the Information Element
            hashInitialiserValue and to psampFiltHashInitializerValue
            in the PSAMP MIB module.";
         reference
           "RFC 5477, Section 8.3.9; RFC 6727, Section 6
            (psampFiltHashInitializerValue)";
       }

       leaf ip-payload-offset {
         type uint64;
         units "octets";
         default '0';
         description
           "IP payload offset indicating the position of the first
            payload byte considered as input to the hash function.

            Default value 0 corresponds to the minimum offset that
            must be configurable according to RFC 5476, Section
            6.5.2.6.

            This parameter corresponds to the Information Element
            hashIPPayloadOffset and to psampFiltHashIpPayloadOffset
            in the PSAMP MIB module.";
         reference
           "RFC 5477, Section 8.3.2; RFC 6727, Section 6
            (psampFiltHashIpPayloadOffset)";
       }

       leaf ip-payload-size {
         type uint64;
         units "octets";
         default '8';
```

```
          description
            "Number of IP payload bytes used as input to the hash
             function, counted from the payload offset.  If the IP
             payload is shorter than the payload range, all available
             payload octets are used as input.

             Default value 8 corresponds to the minimum IP payload
             size that must be configurable according to RFC 5476,
             Section 6.5.2.6.

             This parameter corresponds to the Information Element
             hashIPPayloadSize and to psampFiltHashIpPayloadSize
             in the PSAMP MIB module.";
          reference
            "RFC 5477, Section 8.3.3; RFC 6727, Section 6
             (psampFiltHashIpPayloadSize)";
        }

        leaf digest-output {
          type boolean;
          default 'false';
          description
            "If true, the output from this Selector is included in the
             Packet Report as a packet digest.  Therefore, the configured
             Cache Layout needs to contain a digestHashValue field.

             This parameter corresponds to the Information Element
             hashDigestOutput.";
          reference
            "RFC 5477, Section 8.3.8";
        }

        list selected-range {
          key "name";
          min-elements 1;
          description
            "List of hash function return ranges for which packets are
             selected.";

          leaf name {
            type ipfix:name-type;
            description
              "An arbitrary string which uniquely identifies the
               hash function's selected range.";
          }

          leaf min {
            type uint64;
```

```
          description
            "Beginning of the hash function's selected range.

             This parameter corresponds to the Information Element
             hashSelectedRangeMin and to psampFiltHashSelectedRangeMin
             in the PSAMP MIB module.";
          reference
            "RFC 5477, Section 8.3.6; RFC 6727, Section 6
             (psampFiltHashSelectedRangeMin)";
        }

        leaf max {
          type uint64;
          description
            "End of the hash function's selected range.

             This parameter corresponds to the Information Element
             hashSelectedRangeMax and to psampFiltHashSelectedRangeMax
             in the PSAMP MIB module.";
          reference
            "RFC 5477, Section 8.3.7; RFC 6727, Section 6
             (psampFiltHashSelectedRangeMax)";
        }
      }
    }

    grouping filter-hash-parameters-state {
      description
        "Configuration parameters of a Selector applying hash-based
         Filtering to the packet stream.";
      reference
        "RFC 5475, Section 6.2; RFC 5476, Section 6.5.2.6";

      leaf output-range-min {
        type uint64;
        config false;
        description
          "Beginning of the hash function's potential range.

           This parameter corresponds to the Information Element
           hashOutputRangeMin and to psampFiltHashOutputRangeMin
           in the PSAMP MIB module.";
        reference
          "RFC 5477, Section 8.3.4; RFC 6727, Section 6
           (psampFiltHashOutputRangeMin)";
      }

      leaf output-range-max {
```

```
         type uint64;
         config false;
         description
           "End of the hash function’s potential range.

            This parameter corresponds to the Information Element
            hashOutputRangeMax and to psampFiltHashOutputRangeMax
            in the PSAMP MIB module.";
         reference
           "RFC 5477, Section 8.3.5; RFC 6727, Section 6
            (psampFiltHashOutputRangeMax)";
       }
     }

     grouping selector-parameters {
       description
         "Configuration and state parameters of a Selector.";

       choice method {
         mandatory true;
         description
           "Packet selection method applied by the Selector.";

         case select-all {
           leaf select-all {
             type empty;
             description
               "Method that selects all packets.";
           }
         }

         case samp-count-based {
           container samp-count-based {
             if-feature psamp-samp-count-based;
             description
               "Systematic count-based packet Sampling.";

             uses samp-count-based-parameters;
           }
         }

         case samp-time-based {
           container samp-time-based {
             if-feature psamp-samp-time-based;
             description
               "Systematic time-based packet Sampling.";

             uses samp-time-based-parameters;
```

```
          }
        }

        case samp-rand-out-of-n {
          container samp-rand-out-of-n {
             if-feature psamp-samp-rand-out-of-n;
             description
               "n-out-of-N packet Sampling.";

             uses samp-rand-out-of-n-parameters;
          }
        }

        case samp-uni-prob {
          container samp-uni-prob {
            if-feature psamp-samp-uni-prob;
            description
              "Uniform probabilistic packet Sampling.";

            uses samp-uni-prob-parameters;
          }
        }

        case filter-match {
          container filter-match {
            if-feature psamp-filter-match;
            description
              "Property match Filtering.";

            uses filter-match-parameters;
          }
        }

        case filter-hash {
          container filter-hash {
            if-feature psamp-filter-hash;
            description
              "Hash-based Filtering.";

            uses filter-hash-parameters;
            uses filter-hash-parameters-state;
          }
        }
      }
    }

    grouping selector-parameters-state {
      description
```

```
          "Configuration and state parameters of a Selector.";

        leaf packets-observed {
          type yang:counter64;
          config false;
          description
            "The number of packets observed at the input of the
             Selector.

             If this is the first Selector in the Selection Process,
             this counter corresponds to the total number of packets in
             all Observed Packet Streams at the input of the Selection
             Process.  Otherwise, the counter corresponds to the total
             number of packets at the output of the preceding Selector.

             Discontinuities in the value of this counter can occur at
             re-initialization of the management system, and at other
             times as indicated by the value of
             selectorDiscontinuityTime.

             Note that this parameter corresponds to
             ipfixSelectorStatsPacketsObserved in the IPFIX MIB
             module.";
          reference
            "RFC 6615, Section 8
             (ipfixSelectorStatsPacketsObserved)";
        }

        leaf packets-dropped {
          type yang:counter64;
          config false;
          description
            "The total number of packets discarded by the Selector.

             Discontinuities in the value of this counter can occur at
             re-initialization of the management system, and at other
             times as indicated by the value of
             selectorDiscontinuityTime.

             Note that this parameter corresponds to
             ipfixSelectorStatsPacketsDropped in the IPFIX MIB
             module.";
          reference
            "RFC 6615, Section 8
             (ipfixSelectorStatsPacketsDropped)";
        }

        leaf selector-discontinuity-time {
```

```
        type yang:date-and-time;
        config false;
        description
          "Timestamp of the most recent occasion at which one or more
           of the Selector counters suffered a discontinuity.

           Note that this parameter functionally corresponds to
           ipfixSelectionProcessStatsDiscontinuityTime in the IPFIX
           MIB module. In contrast to
           ipfixSelectionProcessStatsDiscontinuityTime, the time is
           absolute and not relative to sysUpTime.";
        reference
          "RFC 6615, Section 8
           (ipfixSelectionProcessStatsDiscontinuityTime)";
      }
    }

    grouping cache-layout-parameters {
      description
        "Cache Layout parameters used by immediate cache, timeout
         cache, natural cache, and permanent cache.";

      container cache-layout {
        description
          "Cache Layout parameters.";

        list cache-field {
          key "name";
          min-elements 1;
          description
            "Superset of fields that are included in the Packet Reports
             or Flow Records generated by the Cache.";

          leaf name {
            type ipfix:name-type;
            description
              "An arbitrary string which uniquely identifies the
               cache field.";
          }

          choice information-element {
            mandatory true;
            description
              "The Information Element to be added to the template.";
            reference
              "RFC 7012, Section 2; IANA registry for IPFIX
               Entities, http://www.iana.org/assignments/ipfix";
```

```
            case ie-name {
              leaf ie-name {
                type ipfix:ie-name-type;
                description
                  "Name of the Information Element.";
              }
            }

            case ie-id {
              leaf ie-id {
                type ipfix:ie-id-type;
                description
                  "ID of the Information Element.";
              }
            }
          }

          leaf ie-length {
            type uint16;
            units "octets";
            description
              "Length of the field in which the Information Element is
               encoded.  A value of 65535 specifies a variable-length
               Information Element. For Information Elements of integer
               and float type, the field length MAY be set to a smaller
               value than the standard length of the abstract data type
               if the rules of reduced size encoding are fulfilled.

               If not configured by the user, this parameter is set by
               the Monitoring Device.";
            reference
              "RFC 7011, Section 6.2";
          }

          leaf ie-enterprise-number {
            type uint32;
            default '0';
            description
              "If this parameter is zero, the Information Element is
               registered in the IANA registry of IPFIX Information
               Elements.

               If this parameter is configured with a non-zero private
               enterprise number, the Information Element is
               enterprise-specific.

               If the enterprise number is set to 29305, this field
               contains a Reverse Information Element.  In this case,
```

```
                the Cache MUST generate Data Records in accordance to
                RFC 5103.";
            reference
              "RFC 7011; RFC 5103;
               IANA registry for Private Enterprise Numbers,
               http://www.iana.org/assignments/enterprise-numbers;
               IANA registry for IPFIX Entities,
               http://www.iana.org/assignments/ipfix";
          }
        }
      }
    }

    grouping flow-cache-base-parameters {
      description
        "Configuration parameters of a Cache generating Flow Records
         which are common to all Cache types.";

      leaf max-flows {
        type uint32;
        units "flows";
        description
          "This parameter configures the maximum number of Flows in the
           Cache, which is the maximum number of Flows that can be
           measured simultaneously.

           The Monitoring Device MUST ensure that sufficient resources
           are available to store the configured maximum number of
           Flows.

           If the maximum number of Flows is measured, an additional
           Flow can be measured only if an existing entry is removed.
           However, traffic that pertains to existing Flows can
           continue to be measured.";
      }
    }

    grouping flow-permanent-cache-parameters {
      description
        "Configuration parameters of a Permanent Cache generating Flow
         Records.";

      uses flow-cache-base-parameters;

      leaf export-interval {
        type uint32;
        units "seconds";
        description
```

```
           "This parameter configures the interval (in seconds) for
            periodical export of Flow Records.

            If not configured by the user, the Monitoring Device sets
            this parameter.";
       }
     }

     grouping flow-timeout-natural-cache-parameters {
       description
         "Configuration parameters of a Timeout or Natural Cache
          generating Flow Records.";

       uses flow-cache-base-parameters;

       leaf active-timeout {
         type uint32;
         units "seconds";
         description
           "This parameter configures the time in seconds after which a
            Flow is expired even though packets matching this Flow are
            still received by the Cache.

            The parameter value zero indicates infinity, meaning that
            there is no active timeout.

            If not configured by the user, the Monitoring Device sets
            this parameter.

            Note that this parameter corresponds to
            ipfixMeteringProcessCacheActiveTimeout in the IPFIX
            MIB module.";
         reference
           "RFC 6615, Section 8
            (ipfixMeteringProcessCacheActiveTimeout)";
       }

       leaf idle-timeout {
         type uint32;
         units "seconds";
         description
           "This parameter configures the time in seconds after which a
            Flow is expired if no more packets matching this Flow are
            received by the Cache.

            The parameter value zero indicates infinity, meaning that
            there is no idle timeout.
```

```
            If not configured by the user, the Monitoring Device sets
            this parameter.

            Note that this parameter corresponds to
            ipfixMeteringProcessCacheIdleTimeout in the IPFIX
            MIB module.";
         reference
           "RFC 6615, Section 8
            (ipfixMeteringProcessCacheIdleTimeout)";
       }
     }

     grouping flow-cache-parameters-state {
       description
         "State parameters of a Cache generating Flow Records.";

       leaf active-flows {
         type yang:gauge32;
         units "flows";
         config false;
         description
           "The number of Flows currently active in this Cache.

            Note that this parameter corresponds to
            ipfixMeteringProcessCacheActiveFlows in the IPFIX MIB
            module.";
         reference
           "RFC 6615, Section 8
            (ipfixMeteringProcessCacheActiveFlows)";
       }

       leaf unused-cache-entries {
         type yang:gauge32;
         units "flows";
         config false;
         description
           "The number of unused Cache entries in this Cache.

            Note that this parameter corresponds to
            ipfixMeteringProcessCacheUnusedCacheEntries in the IPFIX
            MIB module.";
         reference
           "RFC 6615, Section 8
            (ipfixMeteringProcessCacheUnusedCacheEntries)";
       }
     }

     augment '/ipfix:ipfix' {
```

```
       description
         "Augment IPFIX to add PSAMP.";

       container psamp {
         description
           "Container for PSAMP nodes.";

         list observation-point {
           key "name";
           description
             "Observation Point of the Monitoring Device.";

           leaf name {
             type ipfix:name-type;
             description
               "An arbitrary string which uniquely identifies the
                Observation Point.";
           }

           uses observation-point-parameters;

           leaf-list selection-process {
             type leafref {
               path "/ipfix:ipfix/psamp/selection-process/name";
             }
             description
               "Selection Processes in this list process packets in
                parallel.";
           }

           leaf observation-point-id {
             type uint32;
             config false;
             description
               "Observation Point ID (i.e., the value of the Information
                Element observationPointId) assigned by the Monitoring
                Device.";
             reference
               "IANA registry for IPFIX Entities,
                http://www.iana.org/assignments/ipfix";
           }
         }

         list selection-process {
           key "name";
           description
             "Selection Process of the Monitoring Device.";
```

```
          leaf name {
            type ipfix:name-type;
            description
              "An arbitrary string which uniquely identifies the
               Selectiong Process.";
          }

          list selector {
            key "name";
            min-elements 1;
            ordered-by user;
            description
              "List of Selectors that define the action of the
               Selection Process on a single packet.  The Selectors
               are serially invoked in the same order as they appear
               in this list.";

            leaf name {
              type ipfix:name-type;
              description
                "Name of the selector.";
            }

            uses selector-parameters;

            uses selector-parameters-state;
          }

          leaf cache {
            type leafref {
              path "/ipfix:ipfix/psamp/cache/name";
            }
            description
              "Cache that receives the output of the Selection
               Process.";
          }

          list selection-sequence {
            config false;
            description
              "This list contains the Selection Sequence IDs that are
               assigned by the Monitoring Device to distinguish
               different Selection Sequences passing through the
               Selection Process.

               As Selection Sequence IDs are unique per Observation
               Domain, the corresponding Observation Domain IDs are
               included as well.
```

```
                With this information, it is possible to associate
                Selection Sequence (Statistics) Report Interpretations
                exported according to the PSAMP protocol with a
                Selection Process in the configuration data.";
             reference
               "RFC 5476";

             leaf observation-domain-id {
               type uint32;
               description
                 "Observation Domain ID for which the
                  Selection Sequence ID is assigned.";
             }

             leaf selection-sequence-id {
               type uint64;
               description
                 "Selection Sequence ID used in the Selection
                  Sequence (Statistics) Report Interpretation.";
             }
           }
         }

         list cache {
           key "name";
           description
             "Cache of the Monitoring Device.";

           leaf name {
             type ipfix:name-type;
             description
               "An arbitrary string which uniquely identifies the
                cache.";
           }

           leaf enabled {
             type boolean;
             default "true";
             description
               "If true, this cache is enabled and the specified data is
                able to be exported.";
           }

           choice cache-type {
             mandatory true;
             description
               "Type of Cache and specific parameters.";
```

```
            case immediate-cache {
              container immediate-cache {
                if-feature immediate-cache;
                description
                  "Flow expiration after the first packet; generation
                   of Packet Records.";

                uses cache-layout-parameters;
              }
            }

            case timeout-cache {
              container timeout-cache {
                if-feature timeout-cache;
                description
                  "Flow expiration after active and idle timeout;
                   generation of Flow Records.";

                uses flow-timeout-natural-cache-parameters;
                uses cache-layout-parameters {
                  augment "cache-layout/cache-field" {
                    description
                      "Augment the Cache layout with timeout cache
                       specific configuration.";

                    leaf is-flow-key {
                      when
                        "../ie-enterprise-number != 29305" {
                        description
                          "This parameter is not available for Reverse
                           Information Elements (which have enterprise
                           number 29305).";
                      }
                      type empty;
                      description
                        "If present, this is a flow key.";
                    }
                  }
                }
                uses flow-cache-parameters-state;

              }
            }

            case natural-cache {
              container natural-cache {
                if-feature natural-cache;
                description
```

```
                 "Flow expiration after active and idle timeout, or on
                  natural termination (e.g., TCP FIN or TCP RST) of
                  the Flow; generation of Flow Records.";

               uses flow-timeout-natural-cache-parameters;
               uses cache-layout-parameters {
                 augment "cache-layout/cache-field" {
                   description
                     "Augment the Cache layout with timeout cache
                      specific configuration.";

                   leaf is-flow-key {
                     when
                       "../ie-enterprise-number != 29305" {
                       description
                         "This parameter is not available for Reverse
                          Information Elements (which have enterprise
                          number 29305).";
                     }
                     type empty;
                     description
                       "If present, this is a flow key.";
                   }
                 }
               }
               uses flow-cache-parameters-state;
             }
           }

           case permanent-cache {
             container permanent-cache {
               if-feature permanent-cache;
               description
                 "No flow expiration, periodical export with time
                  interval exportInterval; generation of Flow
                  Records.";
               uses flow-permanent-cache-parameters;
               uses cache-layout-parameters {
                 augment "cache-layout/cache-field" {
                   description
                     "Augment the Cache layout with timeout cache
                      specific configuration.";

                   leaf is-flow-key {
                     when
                       "../ie-enterprise-number != 29305" {
                       description
                         "This parameter is not available for Reverse
```

```
                         Information Elements (which have enterprise
                         number 29305).";
                    }
                    type empty;
                    description
                      "If present, this is a flow key.";
                  }
                }
              }
              uses flow-cache-parameters-state;
            }
          }
        }

        leaf-list exporting-process {
          if-feature ipfix:exporter;
          type leafref {
            path "/ipfix:ipfix"
               + "/ipfix:exporting-process"
               + "/ipfix:name";
          }
          description
            "Records are exported by all Exporting Processes in the
             list.";
        }

        leaf metering-process-id {
          type uint32;
          config false;
          description
            "The identifier of the Metering Process this Cache
             belongs to.

             This parameter corresponds to the Information Element
             meteringProcessId.  Its occurrence helps to associate
             Cache parameters with Metering Process statistics
             exported by the Monitoring Device using the Metering
             Process (Reliability) Statistics Template as
             defined by the IPFIX protocol specification.";
          reference
            "RFC 7011, Sections 4.1 and 4.2;
             IANA registry for IPFIX Entities,
             http://www.iana.org/assignments/ipfix";
        }

        leaf data-records {
          type yang:counter64;
          units "Data Records";
```

```
            config false;
            description
              "The number of Data Records generated by this Cache.

               Discontinuities in the value of this counter can occur
               at re-initialization of the management system, and at
               other times as indicated by the value of
               cacheDiscontinuityTime.

               Note that this parameter corresponds to
               ipfixMeteringProcessDataRecords in the IPFIX MIB
               module.";
            reference
              "RFC 6615, Section 8
               (ipfixMeteringProcessDataRecords)";
          }

          leaf cache-discontinuity-time {
            type yang:date-and-time;
            config false;
            description
              "Timestamp of the most recent occasion at which the
               counter dataRecords suffered a discontinuity.

               Note that this parameter functionally corresponds to
               ipfixMeteringProcessDiscontinuityTime in the IPFIX MIB
               module. In contrast to
               ipfixMeteringProcessDiscontinuityTime, the time is
               absolute and not relative to sysUpTime.";
            reference
              "RFC 6615, Section 8
               (ipfixMeteringProcessDiscontinuityTime)";
          }
        }
      }
    }
  }

  <CODE ENDS>
```

6.3. ietf-ipfix-bulk-data-export

6.3.1. ietf-ipfix-bulk-data-export Module Structure

This document defines the YANG module "ietf-ipfix-bulk-data-export", which has the following tentative structure:

```
module: ietf-ipfix-bulk-data-export
  augment /ipfix:ipfix:
    +--rw bulk-data-export
       +--rw template* [name]
          +--rw name                         ipfix:name-type
          +--rw enabled?                     boolean
          +--rw export-interval?             uint32
          +--rw observation-domain-id?       uint32
          +--rw field-layout
          |     ...
          +--rw exporting-process*
          |       -> /ipfix:ipfix/exporting-process/name
          |       {ipfix:exporter}?
          +--rw (resource-identifier)?
          |     ...
          +--ro data-records?                yang:counter64
          +--ro discontinuity-time?          yang:date-and-time
```

6.3.2. ietf-ipfix-bulk-data-export YANG module

This YANG Module imports typedefs from [RFC6991].

```
<CODE BEGINS> file "ietf-ipfix-bulk-data-export@2018-11-15.yang"

module ietf-ipfix-bulk-data-export {
  yang-version 1.1;

  namespace
    "urn:ietf:params:xml:ns:yang:ietf-ipfix-bulk-data-export";

  prefix ibde;

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

  import ietf-ipfix {
    prefix ipfix;
    reference
      "RFC XXXX: YANG Data Models for the IP Flow Information Export
       (IPFIX) Protocol, Packet Sampling (PSAMP) Protocol, and Bulk
       Data Export";
  }

  organization
    "IETF";
```

```
     contact
       "Web:      TBD
        List:     TBD

        Editor:   Joey Boyd
                  <mailto:joey.boyd@adtran.com>

        Editor:   Marta Seda
                  <mailto:marta.seda@calix.com>";

     // RFC Ed.: replace XXXX with actual RFC numbers and
     // remove this note.

     description
       "This module contains a collection of YANG definitions for the
        management exporting bulk data over IPFIX.

        This data model is designed for the Network Management Datastore
        Architecture defined in RFC 8342.

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

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

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

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

     revision 2020-03-05 {
       description
         "Initial revision.";
       reference
         "RFC XXXX: YANG Data Models for the IP Flow Information Export
                    (IPFIX) Protocol, Packet Sampling (PSAMP) Protocol,
                    and Bulk Data Export";
     }
```

```
     feature bulk-data {
       description
         "If supported, bulk data templates can be configured.";
     }

     typedef resource {
       type instance-identifier {
         require-instance false;
       }
       description
         "A resource from which bulk data will be exported.";
     }

     grouping bulk-data-template-parameters {
       description
         "Field Layout parameters.";

       leaf observation-domain-id {
         type uint32;
         default 0;
         description
           "An identifier of an Observation Domain that is locally
            unique to an Exporting Process (see RFC 7011 Section 3.1).

            Typically, this Information Element is for limiting the
            scope of other Information Elements.

            A value of 0 indicates that no specific Observation Domain
            is identified by this Information Element.";
       }

       container field-layout {
         description
           "Field Layout parameters.";

         list field {
           key name;
           min-elements 1;
           description
             "Superset of statistics field names or special field-names
              (e.g., timestamps, etc) for interpreting statistics that
              are included in the Packet Reports or Flow Records
              generated by the device.";

           leaf name {
             type ipfix:name-type;
             description
               "An arbitrary string which uniquely identifies the
```

```
                 field.";
            }

            choice identifier {
              mandatory true;
              description
                "The Information Element to be added to the template.";

              case ie-name {
                leaf ie-name {
                  type ipfix:ie-name-type;
                  description
                    "Name of the Information Element.";
                }
              }

              case ie-id {
                leaf ie-id {
                  type ipfix:ie-id-type;
                  description
                    "ID of the Information Element.";
                }
              }
            }

            leaf ie-length {
              type uint16;
              units octets;
              description
                "Length of the field in which the Information Element is
                 encoded.  A value of 65535 specifies a variable-length
                 Information Element.  For Information Elements of
                 integer and float type, the field length MAY be set to a
                 smaller value than the standard length of the abstract
                 data type if the rules of reduced size encoding are
                 fulfilled.

                 If not configured by the user, this parameter is set by
                 the Monitoring Device.";
              reference
                "RFC 7011, Section 6.2";
            }

            leaf ie-enterprise-number {
              type uint32;
              default 0;
              description
                "If this parameter is zero, the Information Element is
```

```
                registered in the IANA registry of IPFIX Information
                Elements or unspecified (if the Informational Element is
                not IANA registered).

                If this parameter is configured with a non-zero private
                enterprise number, the Information Element is
                enterprise-specific.";
             reference
               "RFC 7011; RFC 5103;
                IANA registry for Private Enterprise Numbers,
                http://www.iana.org/assignments/enterprise-numbers;
                IANA registry for IPFIX Entities,
                http://www.iana.org/assignments/ipfix";
           }
         }
       }
     }

     augment "/ipfix:ipfix" {
       description
         "Augment IPFIX to add bulk data.";

       container bulk-data-export {
         description
           "Container for bulk data export nodes.";

         list template {
           key name;
           description
             "List of bulk data templates of the Monitoring Device.";

           leaf name {
             type ipfix:name-type;
             description
               "An arbitrary string which uniquely identifies the
                bulk data template.";
           }

           leaf enabled {
             type boolean;
             default "true";
             description
               "If true, this template is enabled and the specified
                data is able to be exported.";
           }

           leaf export-interval {
             type uint32;
```

```
            units "seconds";
            description
              "This parameter configures the interval (in seconds) for
               periodical export of Flow Records.

               If not configured by the user, the Monitoring Device
               sets this parameter.";
          }

          uses bulk-data-template-parameters;

          leaf-list exporting-process {
            if-feature ipfix:exporter;
            type leafref {
              path "/ipfix:ipfix"
                 + "/ipfix:exporting-process"
                 + "/ipfix:name";
            }
            description
              "Records are exported by all Exporting Processes in the
               list.";
          }

          choice resource-identifier {
            description
              "Method to select the resources from which the records
               are to be exported.";

            case resource-instance {
              leaf-list resource-instance {
                type resource;
                description
                  "Records are sourced from all the resources in
                   this list.";
              }
            }
          }

          leaf data-records {
            type yang:counter64;
            units "Data Records";
            config false;
            description
              "The number of Data Records generated for this sampling
               template.

               Discontinuities in the value of this counter can occur
               at re-initialization of the management system, and at
```

```
                other times as indicated by the value of Discontinuity
                Time.";
           }

           leaf discontinuity-time {
             type yang:date-and-time;
             config false;
             description
               "Timestamp of the most recent occasion at which the
                counter data records suffered a discontinuity.";
           }
         }
       }
     }
   }

   <CODE ENDS>
```

7. IANA Considerations

This document registers 3 URIs in the "IETF XML Registry". [RFC3688]. Following the format in RFC 3688, the following registrations have been made.

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

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

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

This document registers 3 YANG modules in the "YANG Module Names" registry. Following the format in [RFC7950], the following have been registered.

Name: ietf-ipfix
Namespace: urn:ietf:params:xml:ns:yang:ietf-ipfix
Prefix: ietf-ipfix
Reference: RFC XXXX: YANG Data Models for the IP Flow Information
           Export (IPFIX) Protocol, Packet Sampling (PSAMP) Protocol,
           and Bulk Data Export

Name: ietf-ipfix-packet-sampling
Namespace: urn:ietf:params:xml:ns:yang:ietf-ipfix-packet-sampling
Prefix: ietf-ipfix-packet-sampling
Reference: RFC XXXX: YANG Data Models for the IP Flow Information Export (IPFIX) Protocol, Packet Sampling (PSAMP) Protocol, and Bulk Data Export

Name: ietf-ipfix-bulk-data-export
Namespace: urn:ietf:params:xml:ns:yang:ietf-ipfix-bulk-data-export
Prefix: ietf-bde
Reference: RFC XXXX: YANG Data Models for the IP Flow Information Export (IPFIX) Protocol, Packet Sampling (PSAMP) Protocol, and Bulk Data Export

8. 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 NETCONF access control model [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., NETCONF 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:

- /ipfix/psamp/observation-point: The configuration parameters in this subtree specify where packets are observed and by which Selection Processes they will be processed. Write access to this subtree allows observing packets at arbitrary interfaces or linecards of the Monitoring Device and may thus lead to the export of sensitive traffic information.

- /ipfix/psamp/selection-process: The configuration parameters in this subtree specify for which packets information will be reported in Packet Reports or Flow Records. Write access to this subtree allows changing the subset of packets for which information will be reported and may thus lead to the export of sensitive traffic information.

- /ipfix/psamp/cache: The configuration parameters in this subtree specify the fields included in Packet Reports or Flow Records. Write access to this subtree allows adding fields which may contain sensitive traffic information, such as IP addresses or parts of the packet payload.

- /ipfix/exporting-process: The configuration parameters in this subtree specify to which Collectors Packet Reports or Flow Records are exported. Write access to this subtree allows exporting potentially sensitive traffic information to illegitimate Collectors. Furthermore, TLS/DTLS parameters can be changed, which may affect the mutual authentication between Exporters and Collectors as well as the encrypted transport of the data.

- /ipfix/collecting-process: The configuration parameters in this subtree may specify that collected Packet Reports and Flow Records are reexported to another Collector or written to a file. Write access to this subtree potentially allows reexporting or storing the sensitive traffic information.

- /ipfix/bulk-data-export/template: The configuration parameters in this subtree specify the fields included in the bulk data export. Write access to this subtree allows adding fields which may cause export of sensitive configuration and/or statistics.

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:

- /ipfix/psamp/observation-point: Parameters in this subtree may be sensitive because they reveal information about the Monitoring Device itself and the network infrastructure.

- /ipfix/psamp/selection-process: Parameters in this subtree may be sensitive because they reveal information about the Monitoring Device itself and the observed traffic. For example, the counters packetsObserved and packetsDropped inferring the number of observed packets.

- /ipfix/psamp/cache: Parameters in this subtree may be sensitive because they reveal information about the Monitoring Device itself and the observed traffic. For example, the counters activeFlows and dataRecords allow inferring the number of measured Flows or packets.

- /ipfix/exporting-process: Parameters in this subtree may be sensitive because they reveal information about the network infrastructure and the outgoing IPFIX Transport Sessions. For example, it discloses the IP addresses of Collectors as well as the deployed TLS/DTLS configuration, which may facilitate the interception of outgoing IPFIX Messages.

- /ipfix/collecting-process: Parameters in this subtree may be sensitive because they reveal information about the network infrastructure and the incoming IPFIX Transport Sessions. For example, it discloses the IP addresses of Exporters as well as the deployed TLS/DTLS configuration, which may facilitate the interception of incoming IPFIX Messages.

9. Acknowledgments

The authors would like to thank Anand Arokiaraj and William Lupton for their contributions towards creation of this document and associated YANG data models.

10. References

10.1. Normative References

[RFC2863] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB", RFC 2863, DOI 10.17487/RFC2863, June 2000, <https://www.rfc-editor.org/info/rfc2863>.

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

[RFC3758] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", RFC 3758, DOI 10.17487/RFC3758, May 2004, <https://www.rfc-editor.org/info/rfc3758>.

[RFC3871] Jones, G., Ed., "Operational Security Requirements for Large Internet Service Provider (ISP) IP Network Infrastructure", RFC 3871, DOI 10.17487/RFC3871, September 2004, <https://www.rfc-editor.org/info/rfc3871>.

[RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <https://www.rfc-editor.org/info/rfc4960>.

[RFC5103] Trammell, B. and E. Boschi, "Bidirectional Flow Export Using IP Flow Information Export (IPFIX)", RFC 5103, DOI 10.17487/RFC5103, January 2008, <https://www.rfc-editor.org/info/rfc5103>.

[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>.

[RFC5475] Zseby, T., Molina, M., Duffield, N., Niccolini, S., and F. Raspall, "Sampling and Filtering Techniques for IP Packet Selection", RFC 5475, DOI 10.17487/RFC5475, March 2009, <https://www.rfc-editor.org/info/rfc5475>.

[RFC5476] Claise, B., Ed., Johnson, A., and J. Quittek, "Packet Sampling (PSAMP) Protocol Specifications", RFC 5476, DOI 10.17487/RFC5476, March 2009, <https://www.rfc-editor.org/info/rfc5476>.

[RFC5477] Dietz, T., Claise, B., Aitken, P., Dressler, F., and G. Carle, "Information Model for Packet Sampling Exports", RFC 5477, DOI 10.17487/RFC5477, March 2009, <https://www.rfc-editor.org/info/rfc5477>.

[RFC5610] Boschi, E., Trammell, B., Mark, L., and T. Zseby, "Exporting Type Information for IP Flow Information Export (IPFIX) Information Elements", RFC 5610, DOI 10.17487/RFC5610, July 2009, <https://www.rfc-editor.org/info/rfc5610>.

[RFC5655] Trammell, B., Boschi, E., Mark, L., Zseby, T., and A. Wagner, "Specification of the IP Flow Information Export (IPFIX) File Format", RFC 5655, DOI 10.17487/RFC5655, October 2009, <https://www.rfc-editor.org/info/rfc5655>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <https://www.rfc-editor.org/info/rfc6347>.

[RFC6526] Claise, B., Aitken, P., Johnson, A., and G. Muenz, "IP Flow Information Export (IPFIX) Per Stream Control Transmission Protocol (SCTP) Stream", RFC 6526, DOI 10.17487/RFC6526, March 2012, <https://www.rfc-editor.org/info/rfc6526>.

[RFC6615] Dietz, T., Ed., Kobayashi, A., Claise, B., and G. Muenz, "Definitions of Managed Objects for IP Flow Information Export", RFC 6615, DOI 10.17487/RFC6615, June 2012, <https://www.rfc-editor.org/info/rfc6615>.

[RFC6727] Dietz, T., Ed., Claise, B., and J. Quittek, "Definitions of Managed Objects for Packet Sampling", RFC 6727, DOI 10.17487/RFC6727, October 2012, <https://www.rfc-editor.org/info/rfc6727>.

[RFC6933] Bierman, A., Romascanu, D., Quittek, J., and M. Chandramouli, "Entity MIB (Version 4)", RFC 6933, DOI 10.17487/RFC6933, May 2013, <https://www.rfc-editor.org/info/rfc6933>.

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

[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>.

[RFC7012] Claise, B., Ed. and B. Trammell, Ed., "Information Model for IP Flow Information Export (IPFIX)", RFC 7012, DOI 10.17487/RFC7012, September 2013, <https://www.rfc-editor.org/info/rfc7012>.

[RFC7119] Claise, B., Kobayashi, A., and B. Trammell, "Operation of the IP Flow Information Export (IPFIX) Protocol on IPFIX Mediators", RFC 7119, DOI 10.17487/RFC7119, February 2014, <https://www.rfc-editor.org/info/rfc7119>.

[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>.

[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>.

[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>.

10.2. Informative References

[BBF.TR-352] Broadband Forum, "Multi-wavelength PON Inter-Channel-Termination Protocol (ICTP) Specification", May 2017, <https://www.broadband-forum.org/technical/download/TR-352.pdf>.

[IANA-ENTERPRISE-NUMBERS] IANA, "Private Enterprise Numbers", <https://www.iana.org/assignments/enterprise-numbers>.

[IANA-IPFIX] IANA, "IP Flow Information Export (IPFIX) Entities", <https://www.iana.org/assignments/ipfix>.

[RFC1141] Mallory, T. and A. Kullberg, "Incremental updating of the Internet checksum", RFC 1141, DOI 10.17487/RFC1141, January 1990, <https://www.rfc-editor.org/info/rfc1141>.

[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>.

[RFC3954] Claise, B., Ed., "Cisco Systems NetFlow Services Export Version 9", RFC 3954, DOI 10.17487/RFC3954, October 2004, <https://www.rfc-editor.org/info/rfc3954>.

[RFC5473] Boschi, E., Mark, L., and B. Claise, "Reducing Redundancy in IP Flow Information Export (IPFIX) and Packet Sampling (PSAMP) Reports", RFC 5473, DOI 10.17487/RFC5473, March 2009, <https://www.rfc-editor.org/info/rfc5473>.

[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>.

[RFC6728] Muenz, G., Claise, B., and P. Aitken, "Configuration Data Model for the IP Flow Information Export (IPFIX) and Packet Sampling (PSAMP) Protocols", RFC 6728, DOI 10.17487/RFC6728, October 2012, <https://www.rfc-editor.org/info/rfc6728>.

[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>.

[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>.

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <https://www.rfc-editor.org/info/rfc8407>.

[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>.

Appendix A. Example: ietf-ipfix Usage

This configuration example configures an IPFIX exporter for a [BBF.TR-352] ICTP Proxy.

```
<ipfix xmlns="urn:ietf:params:xml:ns:yang:ietf-ipfix">
  <exporting-process>
    <name>TR352-exporter</name>
    <enabled>true</enabled>
    <destination>
      <name>ICTP-Proxy1-collector</name>
      <tcp-exporter>
        <source>
          <source-address>192.100.2.1</source-address>
        </source>
        <destination>
          <destination-address>proxy1.sys.com</destination-address>
        </destination>
      </tcp-exporter>
    </destination>
    <options>
      <name>Options 1</name>
      <options-type>extended-type-information</options-type>
      <options-timeout>0</options-timeout>
    </options>
  </exporting-process>
</ipfix>
```


This configuration example configures an IPFIX mediator.

```
<ipfix xmlns="urn:ietf:params:xml:ns:yang:ietf-ipfix">
  <collecting-process>
    <name>OLT-collector</name>
    <tcp-collector>
      <name>myolt-tcp-collector</name>
      <local-address>192.100.2.1</local-address>
    </tcp-collector>
    <exporting-process>OLT-exporter</exporting-process>
  </collecting-process>
  <exporting-process>
    <name>OLT-exporter</name>
    <enabled>true</enabled>
    <destination>
      <name>big-collector</name>
      <tcp-exporter>
        <source>
          <source-address>192.100.2.1</source-address>
        </source>
        <destination>
          <destination-address>collect1.sys.com</destination-address>
        </destination>
      </tcp-exporter>
    </destination>
    <options>
      <name>Options 1</name>
      <options-type>extended-type-information</options-type>
      <options-timeout>0</options-timeout>
    </options>
  </exporting-process>
</ipfix>
```


Appendix B. Example: ietf-ipfix-packet-sampling Usage

This configuration example configures two Observation Points capturing ingress traffic at eth0 and all traffic at eth1. Both Observed Packet Streams enter two different Selection Processes. The first Selection Process implements a Composite Selector of a filter for UDP packets and a random sampler. The second Selection Process implements a Primitive Selector of an ICMP filter. The Selected Packet Streams of both Selection Processes enter the same Cache. The Cache generates a PSAMP Packet Report for every selected packet.

The associated Exporting Process exports to a Collector using PR-SCTP and DTLS. The TLS/DTLS parameters specify that the collector must supply a certificate for the FQDN collector.example.net. Valid certificates from any certification authority will be accepted. As the destination transport port is omitted, the standard IPFIX-over-DTLS port 4740 is used.

The parameters of the Selection Processes are reported as Selection Sequence Report Interpretations and Selector Report Interpretations [RFC5476]. There will be two Selection Sequence Report Interpretations per Selection Process, one for each Observation Point. Selection Sequence Statistics Report Interpretations are exported every 30 seconds (30000 milliseconds).


```
<ipfix xmlns="urn:ietf:params:xml:ns:yang:ietf-ipfix">
  <psamp xmlns=
   "urn:ietf:params:xml:ns:yang:ietf-ipfix-packet-sampling">
    <observation-point>
      <name>OP at eth0 (ingress)</name>
      <observation-domain-id>123</observation-domain-id>
      <interface-ref>eth0</interface-ref>
      <direction>ingress</direction>
      <selection-process>Sampled UDP packets</selection-process>
      <selection-process>ICMP packets</selection-process>
    </observation-point>

    <observation-point>
      <name>OP at eth1</name>
      <observation-domain-id>123</observation-domain-id>
      <interface-ref>eth1</interface-ref>
      <selection-process>Sampled UDP packets</selection-process>
      <selection-process>ICMP packets</selection-process>
    </observation-point>

    <selection-process>
      <name>Sampled UDP packets</name>
      <selector>
        <name>UDP filter</name>
        <filter-match>
          <ie-id>4</ie-id>
          <value>17</value>
        </filter-match>
      </selector>
      <selector>
        <name>10-out-of-100 sampler</name>
        <samp-rand-out-of-n>
          <size>10</size>
          <population>100</population>
        </samp-rand-out-of-n>
      </selector>
      <cache>PSAMP cache</cache>
    </selection-process>

    <selection-process>
      <name>ICMP packets</name>
```

```
          <selector>
            <name>ICMP filter</name>
            <filter-match>
              <ie-id>4</ie-id>
              <value>1</value>
            </filter-match>
          </selector>
          <cache>PSAMP cache</cache>
        </selection-process>

        <cache>
          <name>PSAMP cache</name>
          <immediate-cache>
            <cache-layout>
              <cache-field>
                <name>Field 1: ipHeaderPacketSection</name>
                <ie-id>313</ie-id>
                <ie-length>64</ie-length>
              </cache-field>
              <cache-field>
                <name>Field 2: observationTimeMilliseconds</name>
                <ie-id>322</ie-id>
              </cache-field>
            </cache-layout>
          </immediate-cache>
          <exporting-process>The only exporter</exporting-process>
        </cache>
      </psamp>

      <exporting-process>
        <name>The only exporter</name>
        <enabled>true</enabled>
        <destination>
          <name>PR-SCTP collector</name>
          <sctp-exporter>
            <destination>
              <destination-address>192.0.2.1</destination-address>
            </destination>
            <rate-limit>1000000</rate-limit>
            <timed-reliability>500</timed-reliability>
            <transport-layer-security>
              <remote-subject-fqdn>coll-1.ex.net</remote-subject-fqdn>
            </transport-layer-security>
          </sctp-exporter>
        </destination>
        <options>
          <name>Options 1</name>
          <options-type>selection-sequence</options-type>
```

```
        <options-timeout>0</options-timeout>
      </options>
      <options>
        <name>Options 2</name>
        <options-type>selection-statistics</options-type>
        <options-timeout>30000</options-timeout>
      </options>
    </exporting-process>

  </ipfix>
```

Appendix C. Example: ietf-ipfix-bulk-data-export Usage

The configuration example configures a field-layout template to export Ethernet statistics from eth0 and eth1.

```
<ipfix xmlns="urn:ietf:params:xml:ns:yang:ietf-ipfix"
       xmlns:if="urn:ietf:params:xml:ns:yang:ietf-interfaces">
  <bulk-data-export xmlns=
   "urn:ietf:params:xml:ns:yang:ietf-ipfix-bulk-data-export">

    <template>
      <name>Ethernet Statistics</name>
      <enabled>true</enabled>
      <export-interval>300</export-interval>
      <observation-domain-id>123</observation-domain-id>
      <field-layout>
        <field>
          <name>in-octets</name>
          <ie-id>1001</ie-id>
          <ie-length>4</ie-length>
          <ie-enterprise-number>664</ie-enterprise-number>
        </field>
        <field>
          <name>out-octets</name>
          <ie-id>1002</ie-id>
          <ie-length>4</ie-length>
          <ie-enterprise-number>664</ie-enterprise-number>
        </field>
      </field-layout>
      <exporting-process>The only one</exporting-process>
      <resource-instance>/if:interfaces/if:interface[if:name='eth0']
        </resource-instance>
      <resource-instance>/if:interfaces/if:interface[if:name='eth1']
        </resource-instance>
    </template>
  </bulk-data-export>
  <exporting-process>
    <name>The only one</name>
    <enabled>true</enabled>
    <destination>
      <name>Bulk data collector</name>
      <tcp-exporter>
        <destination>
          <destination-address>192.0.2.2</destination-address>
        </destination>
        <rate-limit>1000000</rate-limit>
        <transport-layer-security>
          <remote-subject-fqdn>coll-2.ex.net</remote-subject-fqdn>
        </transport-layer-security>
      </tcp-exporter>
    </destination>
  </exporting-process>
</ipfix>
```

Appendix D. Tree diagrams

D.1. ietf-ipfix

The complete tree diagram for ietf-ipfix:

```
module: ietf-ipfix
  +--rw ipfix
     +--rw collecting-process* [name] {collector}?
     |  +--rw name                  name-type
     |  +--rw tcp-collector* [name] {tcp-transport}?
     |  |  +--rw name                        name-type
     |  |  +--rw (local-address-method)?
     |  |  |  +--:(local-address)
     |  |  |     +--rw local-address*        inet:host
     |  |  +--rw local-port?                 inet:port-number
     |  |  +--rw transport-layer-security!
     |  |  |  +--rw local-certification-authority-dn*    string
     |  |  |  +--rw local-subject-dn*                    string
     |  |  |  +--rw local-subject-fqdn*
     |  |  |  |       inet:domain-name
     |  |  |  +--rw remote-certification-authority-dn*   string
     |  |  |  +--rw remote-subject-dn*                   string
     |  |  |  +--rw remote-subject-fqdn*
     |  |  |          inet:domain-name
     |  |  +--ro transport-session* [name]
     |  |     +--ro name                                    name-type
     |  |     +--ro ipfix-version?                          uint16
     |  |     +--ro source-address?                         inet:host
     |  |     +--ro destination-address?                    inet:host
     |  |     +--ro source-port?
     |  |     |       inet:port-number
     |  |     +--ro destination-port?
     |  |     |       inet:port-number
     |  |     +--ro status?
     |  |     |       transport-session-status
     |  |     +--ro rate?
     |  |     |       yang:gauge32
     |  |     +--ro bytes?
     |  |     |       yang:counter64
     |  |     +--ro messages?
     |  |     |       yang:counter64
     |  |     +--ro discarded-messages?
     |  |     |       yang:counter64
     |  |     +--ro records?
     |  |     |       yang:counter64
     |  |     +--ro templates?
     |  |     |       yang:counter32
```

```
     |  |     +--ro options-templates?
     |  |     |       yang:counter32
     |  |     +--ro transport-session-start-time?
     |  |     |       yang:date-and-time
     |  |     +--ro transport-session-discontinuity-time?
     |  |     |       yang:date-and-time
     |  |     +--ro template* [name]
     |  |        +--ro name                          name-type
     |  |        +--ro observation-domain-id?        uint32
     |  |        +--ro template-id?                  uint16
     |  |        +--ro set-id?                       uint16
     |  |        +--ro access-time?
     |  |        |       yang:date-and-time
     |  |        +--ro template-data-records?        yang:counter64
     |  |        +--ro template-discontinuity-time?
     |  |        |       yang:date-and-time
     |  |        +--ro field* [name]
     |  |           +--ro name                    name-type
     |  |           +--ro ie-id?                  ie-id-type
     |  |           +--ro ie-length?              uint16
     |  |           +--ro ie-enterprise-number?   uint32
     |  |           +--ro is-flow-key?            empty
     |  |           +--ro is-scope?               empty
     |  +--rw udp-collector* [name] {udp-transport}?
     |  |  +--rw name                            name-type
     |  |  +--rw (local-address-method)?
     |  |  |  +--:(local-address)
     |  |  |     +--rw local-address*           inet:host
     |  |  +--rw local-port?                     inet:port-number
     |  |  +--rw template-life-time?             uint32
     |  |  +--rw options-template-life-time?     uint32
     |  |  +--rw template-life-packet?           uint32
     |  |  +--rw options-template-life-packet?   uint32
     |  |  +--rw maximum-reordering-delay?       uint32
     |  |  +--rw transport-layer-security!
     |  |  |  +--rw local-certification-authority-dn*    string
     |  |  |  +--rw local-subject-dn*                    string
     |  |  |  +--rw local-subject-fqdn*
     |  |  |  |       inet:domain-name
     |  |  |  +--rw remote-certification-authority-dn*   string
     |  |  |  +--rw remote-subject-dn*                   string
     |  |  |  +--rw remote-subject-fqdn*
     |  |  |          inet:domain-name
     |  |  +--ro transport-session* [name]
     |  |     +--ro name                                    name-type
     |  |     +--ro ipfix-version?                          uint16
     |  |     +--ro source-address?                         inet:host
     |  |     +--ro destination-address?                    inet:host
```

```
     |  |     +--ro source-port?
     |  |     |       inet:port-number
     |  |     +--ro destination-port?
     |  |     |       inet:port-number
     |  |     +--ro status?
     |  |     |       transport-session-status
     |  |     +--ro rate?
     |  |     |       yang:gauge32
     |  |     +--ro bytes?
     |  |     |       yang:counter64
     |  |     +--ro messages?
     |  |     |       yang:counter64
     |  |     +--ro discarded-messages?
     |  |     |       yang:counter64
     |  |     +--ro records?
     |  |     |       yang:counter64
     |  |     +--ro templates?
     |  |     |       yang:counter32
     |  |     +--ro options-templates?
     |  |     |       yang:counter32
     |  |     +--ro transport-session-start-time?
     |  |     |       yang:date-and-time
     |  |     +--ro transport-session-discontinuity-time?
     |  |     |       yang:date-and-time
     |  |     +--ro template* [name]
     |  |        +--ro name                             name-type
     |  |        +--ro observation-domain-id?          uint32
     |  |        +--ro template-id?                    uint16
     |  |        +--ro set-id?                         uint16
     |  |        +--ro access-time?
     |  |        |       yang:date-and-time
     |  |        +--ro template-data-records?          yang:counter64
     |  |        +--ro template-discontinuity-time?
     |  |        |       yang:date-and-time
     |  |        +--ro field* [name]
     |  |           +--ro name                      name-type
     |  |           +--ro ie-id?                    ie-id-type
     |  |           +--ro ie-length?                uint16
     |  |           +--ro ie-enterprise-number?     uint32
     |  |           +--ro is-flow-key?              empty
     |  |           +--ro is-scope?                 empty
     |  +--rw sctp-collector* [name] {sctp-transport}?
     |  |  +--rw name                          name-type
     |  |  +--rw (local-address-method)?
     |  |  |  +--:(local-address)
     |  |  |     +--rw local-address*        inet:host
     |  |  +--rw local-port?                   inet:port-number
     |  |  +--rw maximum-reordering-delay?   uint32
```

```
     |  |  +--rw transport-layer-security!
     |  |  |  +--rw local-certification-authority-dn*    string
     |  |  |  +--rw local-subject-dn*                     string
     |  |  |  +--rw local-subject-fqdn*
     |  |  |  |       inet:domain-name
     |  |  |  +--rw remote-certification-authority-dn*   string
     |  |  |  +--rw remote-subject-dn*                    string
     |  |  |  +--rw remote-subject-fqdn*
     |  |  |          inet:domain-name
     |  |  +--ro transport-session* [name]
     |  |     +--ro name                                      name-type
     |  |     +--ro sctp-association-id?                      uint32
     |  |     +--ro ipfix-version?                            uint16
     |  |     +--ro source-address?                           inet:host
     |  |     +--ro destination-address?                      inet:host
     |  |     +--ro source-port?
     |  |     |       inet:port-number
     |  |     +--ro destination-port?
     |  |     |       inet:port-number
     |  |     +--ro status?
     |  |     |       transport-session-status
     |  |     +--ro rate?
     |  |     |       yang:gauge32
     |  |     +--ro bytes?
     |  |     |       yang:counter64
     |  |     +--ro messages?
     |  |     |       yang:counter64
     |  |     +--ro discarded-messages?
     |  |     |       yang:counter64
     |  |     +--ro records?
     |  |     |       yang:counter64
     |  |     +--ro templates?
     |  |     |       yang:counter32
     |  |     +--ro options-templates?
     |  |     |       yang:counter32
     |  |     +--ro transport-session-start-time?
     |  |     |       yang:date-and-time
     |  |     +--ro transport-session-discontinuity-time?
     |  |     |       yang:date-and-time
     |  |     +--ro template* [name]
     |  |        +--ro name                            name-type
     |  |        +--ro observation-domain-id?         uint32
     |  |        +--ro template-id?                   uint16
     |  |        +--ro set-id?                        uint16
     |  |        +--ro access-time?
     |  |        |       yang:date-and-time
     |  |        +--ro template-data-records?         yang:counter64
     |  |        +--ro template-discontinuity-time?
```

```
     |  |        |       yang:date-and-time
     |  |        +--ro field* [name]
     |  |           +--ro name                      name-type
     |  |           +--ro ie-id?                    ie-id-type
     |  |           +--ro ie-length?                uint16
     |  |           +--ro ie-enterprise-number?     uint32
     |  |           +--ro is-flow-key?              empty
     |  |           +--ro is-scope?                 empty
     |  +--rw file-reader* [name] {file-reader}?
     |  |  +--rw name                  name-type
     |  |  +--rw file                  inet:uri
     |  |  +--ro file-reader-state
     |  |     +--ro bytes?                              yang:counter64
     |  |     +--ro messages?                           yang:counter64
     |  |     +--ro records?                            yang:counter64
     |  |     +--ro templates?                          yang:counter32
     |  |     +--ro options-templates?                  yang:counter32
     |  |     +--ro file-reader-discontinuity-time?
     |  |     |       yang:date-and-time
     |  |     +--ro template* [name]
     |  |        +--ro name                             name-type
     |  |        +--ro observation-domain-id?           uint32
     |  |        +--ro template-id?                     uint16
     |  |        +--ro set-id?                          uint16
     |  |        +--ro access-time?
     |  |        |       yang:date-and-time
     |  |        +--ro template-data-records?           yang:counter64
     |  |        +--ro template-discontinuity-time?
     |  |        |       yang:date-and-time
     |  |        +--ro field* [name]
     |  |           +--ro name                      name-type
     |  |           +--ro ie-id?                    ie-id-type
     |  |           +--ro ie-length?                uint16
     |  |           +--ro ie-enterprise-number?     uint32
     |  |           +--ro is-flow-key?              empty
     |  |           +--ro is-scope?                 empty
     |  +--rw exporting-process*   -> /ipfix/exporting-process/name
     |          {exporter}?
     +--rw exporting-process* [name] {exporter}?
        +--rw name                     name-type
        +--rw enabled?                 boolean
        +--rw export-mode?             identityref
        +--rw destination* [name]
        |  +--rw name                    name-type
        |  +--rw (destination-parameters)
        |     +--:(tcp-exporter)
        |     |  +--rw tcp-exporter {tcp-transport}?
        |     |     +--rw ipfix-version?            uint16
```

```
     |        |      +--rw source
     |        |      |  +--rw (source-method)?
     |        |      |     +--:(interface-ref)
     |        |      |     |  +--rw interface-ref?    if:interface-ref
     |        |      |     +--:(if-index) {if-mib}?
     |        |      |     |  +--rw if-index?         uint32
     |        |      |     +--:(if-name) {if-mib}?
     |        |      |     |  +--rw if-name?          string
     |        |      |     +--:(source-address)
     |        |      |        +--rw source-address?   inet:host
     |        |      +--rw destination
     |        |      |  +--rw (destination-method)
     |        |      |     +--:(destination-address)
     |        |      |        +--rw destination-address?   inet:host
     |        |      +--rw destination-port?
     |        |      |       inet:port-number
     |        |      +--rw send-buffer-size?           uint32
     |        |      +--rw rate-limit?                 uint32
     |        |      +--rw connection-timeout?         uint32
     |        |      +--rw retry-schedule?             uint32
     |        |      +--rw transport-layer-security!
     |        |      |  +--rw local-certification-authority-dn*
     |        |      |  |       string
     |        |      |  +--rw local-subject-dn*
     |        |      |  |       string
     |        |      |  +--rw local-subject-fqdn*
     |        |      |  |       inet:domain-name
     |        |      |  +--rw remote-certification-authority-dn*
     |        |      |  |       string
     |        |      |  +--rw remote-subject-dn*
     |        |      |  |       string
     |        |      |  +--rw remote-subject-fqdn*
     |        |      |          inet:domain-name
     |        |      +--ro transport-session
     |        |         +--ro ipfix-version?
     |        |         |       uint16
     |        |         +--ro source-address?
     |        |         |       inet:host
     |        |         +--ro destination-address?
     |        |         |       inet:host
     |        |         +--ro source-port?
     |        |         |       inet:port-number
     |        |         +--ro destination-port?
     |        |         |       inet:port-number
     |        |         +--ro status?
     |        |         |       transport-session-status
     |        |         +--ro rate?
     |        |         |       yang:gauge32
```

```
     |     |        +--ro bytes?
     |     |        |       yang:counter64
     |     |        +--ro messages?
     |     |        |       yang:counter64
     |     |        +--ro discarded-messages?
     |     |        |       yang:counter64
     |     |        +--ro records?
     |     |        |       yang:counter64
     |     |        +--ro templates?
     |     |        |       yang:counter32
     |     |        +--ro options-templates?
     |     |        |       yang:counter32
     |     |        +--ro transport-session-start-time?
     |     |        |       yang:date-and-time
     |     |        +--ro transport-session-discontinuity-time?
     |     |        |       yang:date-and-time
     |     |        +--ro template* [name]
     |     |           +--ro name
     |     |           |       name-type
     |     |           +--ro observation-domain-id?         uint32
     |     |           +--ro template-id?                   uint16
     |     |           +--ro set-id?                        uint16
     |     |           +--ro access-time?
     |     |           |       yang:date-and-time
     |     |           +--ro template-data-records?
     |     |           |       yang:counter64
     |     |           +--ro template-discontinuity-time?
     |     |           |       yang:date-and-time
     |     |           +--ro field* [name]
     |     |              +--ro name                     name-type
     |     |              +--ro ie-id?                   ie-id-type
     |     |              +--ro ie-length?               uint16
     |     |              +--ro ie-enterprise-number?    uint32
     |     |              +--ro is-flow-key?             empty
     |     |              +--ro is-scope?                empty
     |     +--:(udp-exporter)
     |     |  +--rw udp-exporter {udp-transport}?
     |     |     +--rw ipfix-version?                          uint16
     |     |     +--rw source
     |     |     |  +--rw (source-method)?
     |     |     |     +--:(interface-ref)
     |     |     |     |  +--rw interface-ref?    if:interface-ref
     |     |     |     +--:(if-index) {if-mib}?
     |     |     |     |  +--rw if-index?         uint32
     |     |     |     +--:(if-name) {if-mib}?
     |     |     |     |  +--rw if-name?          string
     |     |     |     +--:(source-address)
     |     |     |        +--rw source-address?   inet:host
```

```
           |     |     +--rw destination
           |     |     |  +--rw (destination-method)
           |     |     |     +--:(destination-address)
           |     |     |        +--rw destination-address?   inet:host
           |     |     +--rw destination-port?
           |     |     |       inet:port-number
           |     |     +--rw send-buffer-size?                   uint32
           |     |     +--rw rate-limit?                         uint32
           |     |     +--rw maximum-packet-size?                uint16
           |     |     +--rw template-refresh-timeout?           uint32
           |     |     +--rw options-template-refresh-timeout?   uint32
           |     |     +--rw template-refresh-packet?            uint32
           |     |     +--rw options-template-refresh-packet?    uint32
           |     |     +--rw transport-layer-security!
           |     |     |  +--rw local-certification-authority-dn*
           |     |     |  |       string
           |     |     |  +--rw local-subject-dn*
           |     |     |  |       string
           |     |     |  +--rw local-subject-fqdn*
           |     |     |  |       inet:domain-name
           |     |     |  +--rw remote-certification-authority-dn*
           |     |     |  |       string
           |     |     |  +--rw remote-subject-dn*
           |     |     |  |       string
           |     |     |  +--rw remote-subject-fqdn*
           |     |     |          inet:domain-name
           |     |     +--ro transport-session
           |     |        +--ro ipfix-version?
           |     |        |       uint16
           |     |        +--ro source-address?
           |     |        |       inet:host
           |     |        +--ro destination-address?
           |     |        |       inet:host
           |     |        +--ro source-port?
           |     |        |       inet:port-number
           |     |        +--ro destination-port?
           |     |        |       inet:port-number
           |     |        +--ro status?
           |     |        |       transport-session-status
           |     |        +--ro rate?
           |     |        |       yang:gauge32
           |     |        +--ro bytes?
           |     |        |       yang:counter64
           |     |        +--ro messages?
           |     |        |       yang:counter64
           |     |        +--ro discarded-messages?
           |     |        |       yang:counter64
           |     |        +--ro records?
```

```
     |     |        |       yang:counter64
     |     |        +--ro templates?
     |     |        |       yang:counter32
     |     |        +--ro options-templates?
     |     |        |       yang:counter32
     |     |        +--ro transport-session-start-time?
     |     |        |       yang:date-and-time
     |     |        +--ro transport-session-discontinuity-time?
     |     |        |       yang:date-and-time
     |     |        +--ro template* [name]
     |     |           +--ro name
     |     |           |       name-type
     |     |           +--ro observation-domain-id?         uint32
     |     |           +--ro template-id?                   uint16
     |     |           +--ro set-id?                        uint16
     |     |           +--ro access-time?
     |     |           |       yang:date-and-time
     |     |           +--ro template-data-records?
     |     |           |       yang:counter64
     |     |           +--ro template-discontinuity-time?
     |     |           |       yang:date-and-time
     |     |           +--ro field* [name]
     |     |              +--ro name                    name-type
     |     |              +--ro ie-id?                  ie-id-type
     |     |              +--ro ie-length?              uint16
     |     |              +--ro ie-enterprise-number?   uint32
     |     |              +--ro is-flow-key?            empty
     |     |              +--ro is-scope?               empty
     |     +--:(sctp-exporter)
     |     |  +--rw sctp-exporter {sctp-transport}?
     |     |     +--rw ipfix-version?             uint16
     |     |     +--rw source
     |     |     |  +--rw (source-method)?
     |     |     |     +--:(interface-ref)
     |     |     |     |  +--rw interface-ref?    if:interface-ref
     |     |     |     +--:(if-index) {if-mib}?
     |     |     |     |  +--rw if-index?         uint32
     |     |     |     +--:(if-name) {if-mib}?
     |     |     |     |  +--rw if-name?          string
     |     |     |     +--:(source-address)
     |     |     |        +--rw source-address*   inet:host
     |     |     +--rw destination
     |     |     |  +--rw (destination-method)
     |     |     |     +--:(destination-address)
     |     |     |        +--rw destination-address*   inet:host
     |     |     +--rw destination-port?
     |     |     |       inet:port-number
     |     |     +--rw send-buffer-size?           uint32
```

```
           |     |     +--rw rate-limit?                 uint32
           |     |     +--rw timed-reliability?          uint32
           |     |     +--rw association-timeout?        uint32
           |     |     +--rw transport-layer-security!
           |     |     |  +--rw local-certification-authority-dn*
           |     |     |  |       string
           |     |     |  +--rw local-subject-dn*
           |     |     |  |       string
           |     |     |  +--rw local-subject-fqdn*
           |     |     |  |       inet:domain-name
           |     |     |  +--rw remote-certification-authority-dn*
           |     |     |  |       string
           |     |     |  +--rw remote-subject-dn*
           |     |     |  |       string
           |     |     |  +--rw remote-subject-fqdn*
           |     |     |          inet:domain-name
           |     |     +--ro transport-session
           |     |        +--ro sctp-association-id?
           |     |        |       uint32
           |     |        +--ro ipfix-version?
           |     |        |       uint16
           |     |        +--ro source-address?
           |     |        |       inet:host
           |     |        +--ro destination-address?
           |     |        |       inet:host
           |     |        +--ro source-port?
           |     |        |       inet:port-number
           |     |        +--ro destination-port?
           |     |        |       inet:port-number
           |     |        +--ro status?
           |     |        |       transport-session-status
           |     |        +--ro rate?
           |     |        |       yang:gauge32
           |     |        +--ro bytes?
           |     |        |       yang:counter64
           |     |        +--ro messages?
           |     |        |       yang:counter64
           |     |        +--ro discarded-messages?
           |     |        |       yang:counter64
           |     |        +--ro records?
           |     |        |       yang:counter64
           |     |        +--ro templates?
           |     |        |       yang:counter32
           |     |        +--ro options-templates?
           |     |        |       yang:counter32
           |     |        +--ro transport-session-start-time?
           |     |        |       yang:date-and-time
           |     |        +--ro transport-session-discontinuity-time?
```

```
           |     |        |       yang:date-and-time
           |     |        +--ro template* [name]
           |     |           +--ro name
           |     |           |       name-type
           |     |           +--ro observation-domain-id?         uint32
           |     |           +--ro template-id?                   uint16
           |     |           +--ro set-id?                        uint16
           |     |           +--ro access-time?
           |     |           |       yang:date-and-time
           |     |           +--ro template-data-records?
           |     |           |       yang:counter64
           |     |           +--ro template-discontinuity-time?
           |     |           |       yang:date-and-time
           |     |           +--ro field* [name]
           |     |              +--ro name                    name-type
           |     |              +--ro ie-id?                  ie-id-type
           |     |              +--ro ie-length?              uint16
           |     |              +--ro ie-enterprise-number?   uint32
           |     |              +--ro is-flow-key?            empty
           |     |              +--ro is-scope?               empty
           |     +--:(file-writer)
           |        +--rw file-writer {file-writer}?
           |           +--rw ipfix-version?       uint16
           |           +--rw file                 inet:uri
           |           +--ro file-writer-state
           |              +--ro bytes?
           |              |       yang:counter64
           |              +--ro messages?
           |              |       yang:counter64
           |              +--ro discarded-messages?
           |              |       yang:counter64
           |              +--ro records?
           |              |       yang:counter64
           |              +--ro templates?
           |              |       yang:counter32
           |              +--ro options-templates?
           |              |       yang:counter32
           |              +--ro file-writer-discontinuity-time?
           |              |       yang:date-and-time
           |              +--ro template* [name]
           |                 +--ro name
           |                 |       name-type
           |                 +--ro observation-domain-id?         uint32
           |                 +--ro template-id?                   uint16
           |                 +--ro set-id?                        uint16
           |                 +--ro access-time?
           |                 |       yang:date-and-time
           |                 +--ro template-data-records?
```

```
           |                 |       yang:counter64
           |                 +--ro template-discontinuity-time?
           |                 |       yang:date-and-time
           |                 +--ro field* [name]
           |                    +--ro name                     name-type
           |                    +--ro ie-id?                   ie-id-type
           |                    +--ro ie-length?               uint16
           |                    +--ro ie-enterprise-number?    uint32
           |                    +--ro is-flow-key?             empty
           |                    +--ro is-scope?                empty
           +--rw options* [name]
           |  +--rw name               name-type
           |  +--rw options-type       identityref
           |  +--rw options-timeout?   uint32
           +--ro exporting-process-id?   uint32
```

D.2. ietf-ipfix-packet-sampling

The complete tree diagram for ietf-ipfix-packet-sampling:

```
   module: ietf-ipfix-packet-sampling
     augment /ipfix:ipfix:
       +--rw psamp
          +--rw observation-point* [name]
          |  +--rw name                      ipfix:name-type
          |  +--rw observation-domain-id    uint32
          |  +--rw interface-ref*            if:interface-ref
          |  +--rw if-name*                  if-name-type {if-mib}?
          |  +--rw if-index*                 uint32 {if-mib}?
          |  +--rw hardware-ref*             hardware-ref
          |  +--rw ent-physical-name*        string {entity-mib}?
          |  +--rw ent-physical-index*       uint32 {entity-mib}?
          |  +--rw direction?                direction
          |  +--rw selection-process*
          |  |       -> /ipfix:ipfix/psamp/selection-process/name
          |  +--ro observation-point-id?    uint32
          +--rw selection-process* [name]
          |  +--rw name                   ipfix:name-type
          |  +--rw selector* [name]
          |  |  +--rw name                            ipfix:name-type
          |  |  +--rw (method)
          |  |  |  +--:(select-all)
          |  |  |  |  +--rw select-all?             empty
          |  |  |  +--:(samp-count-based)
          |  |  |  |  +--rw samp-count-based {psamp-samp-count-based}?
          |  |  |  |     +--rw packet-interval    uint32
          |  |  |  |     +--rw packet-space       uint32
          |  |  |  +--:(samp-time-based)
```

```
     |  |  |  |  +--rw samp-time-based {psamp-samp-time-based}?
     |  |  |  |     +--rw time-interval    uint32
     |  |  |  |     +--rw time-space       uint32
     |  |  |  +--:(samp-rand-out-of-n)
     |  |  |  |  +--rw samp-rand-out-of-n
     |  |  |  |          {psamp-samp-rand-out-of-n}?
     |  |  |  |     +--rw size          uint32
     |  |  |  |     +--rw population    uint32
     |  |  |  +--:(samp-uni-prob)
     |  |  |  |  +--rw samp-uni-prob {psamp-samp-uni-prob}?
     |  |  |  |     +--rw probability    decimal64
     |  |  |  +--:(filter-match)
     |  |  |  |  +--rw filter-match {psamp-filter-match}?
     |  |  |  |     +--rw (information-element)
     |  |  |  |     |  +--:(ie-name)
     |  |  |  |     |  |  +--rw ie-name?
     |  |  |  |     |  |          ipfix:ie-name-type
     |  |  |  |     |  +--:(ie-id)
     |  |  |  |     |     +--rw ie-id?            ipfix:ie-id-type
     |  |  |  |     +--rw ie-enterprise-number?   uint32
     |  |  |  |     +--rw value                   string
     |  |  |  +--:(filter-hash)
     |  |  |     +--rw filter-hash {psamp-filter-hash}?
     |  |  |        +--rw hash-function?       identityref
     |  |  |        +--rw initializer-value?   uint64
     |  |  |        +--rw ip-payload-offset?   uint64
     |  |  |        +--rw ip-payload-size?     uint64
     |  |  |        +--rw digest-output?       boolean
     |  |  |        +--rw selected-range* [name]
     |  |  |        |  +--rw name    ipfix:name-type
     |  |  |        |  +--rw min?    uint64
     |  |  |        |  +--rw max?    uint64
     |  |  |        +--ro output-range-min?    uint64
     |  |  |        +--ro output-range-max?    uint64
     |  |  +--ro packets-observed?             yang:counter64
     |  |  +--ro packets-dropped?              yang:counter64
     |  |  +--ro selector-discontinuity-time?   yang:date-and-time
     |  +--rw cache?
     |  |       -> /ipfix:ipfix/psamp/cache/name
     |  +--ro selection-sequence* []
     |     +--ro observation-domain-id?   uint32
     |     +--ro selection-sequence-id?   uint64
     +--rw cache* [name]
        +--rw name                       ipfix:name-type
        +--rw enabled?                   boolean
        +--rw (cache-type)
        |  +--:(immediate-cache)
        |  |  +--rw immediate-cache {immediate-cache}?
```

```
             |  |     +--rw cache-layout
             |  |        +--rw cache-field* [name]
             |  |           +--rw name
             |  |           |       ipfix:name-type
             |  |           +--rw (information-element)
             |  |           |  +--:(ie-name)
             |  |           |  |  +--rw ie-name?
             |  |           |  |          ipfix:ie-name-type
             |  |           |  +--:(ie-id)
             |  |           |     +--rw ie-id?
             |  |           |             ipfix:ie-id-type
             |  |           +--rw ie-length?              uint16
             |  |           +--rw ie-enterprise-number?   uint32
             |  +--:(timeout-cache)
             |  |  +--rw timeout-cache {timeout-cache}?
             |  |     +--rw max-flows?              uint32
             |  |     +--rw active-timeout?         uint32
             |  |     +--rw idle-timeout?           uint32
             |  |     +--rw cache-layout
             |  |     |  +--rw cache-field* [name]
             |  |     |     +--rw name
             |  |     |     |       ipfix:name-type
             |  |     |     +--rw (information-element)
             |  |     |     |  +--:(ie-name)
             |  |     |     |  |  +--rw ie-name?
             |  |     |     |  |          ipfix:ie-name-type
             |  |     |     |  +--:(ie-id)
             |  |     |     |     +--rw ie-id?
             |  |     |     |             ipfix:ie-id-type
             |  |     |     +--rw ie-length?              uint16
             |  |     |     +--rw ie-enterprise-number?   uint32
             |  |     |     +--rw is-flow-key?            empty
             |  |     +--ro active-flows?           yang:gauge32
             |  |     +--ro unused-cache-entries?   yang:gauge32
             |  +--:(natural-cache)
             |  |  +--rw natural-cache {natural-cache}?
             |  |     +--rw max-flows?              uint32
             |  |     +--rw active-timeout?         uint32
             |  |     +--rw idle-timeout?           uint32
             |  |     +--rw cache-layout
             |  |     |  +--rw cache-field* [name]
             |  |     |     +--rw name
             |  |     |     |       ipfix:name-type
             |  |     |     +--rw (information-element)
             |  |     |     |  +--:(ie-name)
             |  |     |     |  |  +--rw ie-name?
             |  |     |     |  |          ipfix:ie-name-type
             |  |     |     |  +--:(ie-id)
```

```
     |  |     |     |     +--rw ie-id?
     |  |     |     |             ipfix:ie-id-type
     |  |     |     +--rw ie-length?              uint16
     |  |     |     +--rw ie-enterprise-number?   uint32
     |  |     |     +--rw is-flow-key?            empty
     |  |     +--ro active-flows?           yang:gauge32
     |  |     +--ro unused-cache-entries?   yang:gauge32
     |  +--:(permanent-cache)
     |     +--rw permanent-cache {permanent-cache}?
     |        +--rw max-flows?              uint32
     |        +--rw export-interval?        uint32
     |        +--rw cache-layout
     |        |  +--rw cache-field* [name]
     |        |     +--rw name
     |        |     |       ipfix:name-type
     |        |     +--rw (information-element)
     |        |     |  +--:(ie-name)
     |        |     |  |  +--rw ie-name?
     |        |     |  |          ipfix:ie-name-type
     |        |     |  +--:(ie-id)
     |        |     |     +--rw ie-id?
     |        |     |             ipfix:ie-id-type
     |        |     +--rw ie-length?              uint16
     |        |     +--rw ie-enterprise-number?   uint32
     |        |     +--rw is-flow-key?            empty
     |        +--ro active-flows?           yang:gauge32
     |        +--ro unused-cache-entries?   yang:gauge32
     +--rw exporting-process*
     |       -> /ipfix:ipfix/exporting-process/name
     |       {ipfix:exporter}?
     +--ro metering-process-id?       uint32
     +--ro data-records?              yang:counter64
     +--ro cache-discontinuity-time?   yang:date-and-time
```

D.3. ietf-ipfix-bulk-data-export

The complete tree diagram for ietf-ipfix-bulk-data-export:

```
module: ietf-ipfix-bulk-data-export
  augment /ipfix:ipfix:
    +--rw bulk-data-export
       +--rw template* [name]
          +--rw name                         ipfix:name-type
          +--rw enabled?                     boolean
          +--rw export-interval?             uint32
          +--rw observation-domain-id?       uint32
          +--rw field-layout
          |  +--rw field* [name]
          |     +--rw name                      ipfix:name-type
          |     +--rw (identifier)
          |     |  +--:(ie-name)
          |     |  |  +--rw ie-name?            ipfix:ie-name-type
          |     |  +--:(ie-id)
          |     |     +--rw ie-id?              ipfix:ie-id-type
          |     +--rw ie-length?                uint16
          |     +--rw ie-enterprise-number?     uint32
          +--rw exporting-process*
          |       -> /ipfix:ipfix/exporting-process/name
          |       {ipfix:exporter}?
          +--rw (resource-identifier)?
          |  +--:(resource-instance)
          |     +--rw resource-instance*   resource
          +--ro data-records?                yang:counter64
          +--ro discontinuity-time?          yang:date-and-time
```

Authors' Addresses

Joey Boyd
ADTRAN

Email: joey.boyd@adtran.com

Marta Seda
Calix

Email: marta.seda@calix.com

OPSAWG Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 9, 2019

R. Even
B. Wu
Q. Wu
Huawei
Y. Cheng
China Unicom

March 8, 2019

YANG Data Model for Composed VPN Service Delivery
draft-evenwu-opsawg-yang-composed-vpn-03

Abstract

This document defines a YANG data model that can be used by a network operator to configure a VPN service that spans multiple administrative domains and that is constructed from component VPNs in each of those administrative domains. The component VPNs may be L2VPN or L3VPN or a mixture of the two. This model is intended to be instantiated at the management system to deliver the end to end service (i.e., performing service provision and activation functions at different levels through a unified interface).

The model is not a configuration model to be used directly on network elements. This model provides an abstracted common view of VPN service configuration components segmented at different layer and administrative domain. It is up to a management system to take this as an input and generate specific configurations models to configure the different network elements within each administrative domain to deliver the service. How configuration of network elements is done is out of scope of the document.

Status of This Memo

Table of Contents

1. Introduction

In some cases, a VPN service needs to span different administrative domains. This will usually arise when there are internal administrative boundaries within a single Service Provider's (SP's) network. The boundaries may reflect geographic dispersal or functional decomposition, e.g., access, metro, backhaul, core, and data center.

In particular, the different domains could deploy Layer 2 or Layer 3 technologies or both, and could establish layer-dependent connectivity services. For example, some SPs offer a L2VPN service in the metro access network and extend it across the core network as an IP VPN to provide end-to-end BGP IP VPN services to their enterprise customers.

Some SPs integrate Mobile Backhaul Network and Core networks to provide mobile broadband services. These require stitching multiple layer-dependent connectivity services at different administrative domain boundaries.

This document defines a YANG data model that can be used by a network operator to construct an end-to-end service across multiple administrative domains. This service is delivered by provisioning VPN services utilising Layer 2 or Layer 3 technologies in each domain.

This model is intended to be instantiated at the management system to deliver the overall service per [RFC8309]. It is not a configuration model to be used directly on network elements. This model provides an abstracted common view of VPN service configuration components segmented at different layers and administrative domains. It is up to a management system to take this as an input and generate specific configurations models to configure the different network elements within each administrative domain to deliver the service. How configuration of network elements is done is out of scope of the document. END

1.1. Terminology

The following terms are defined in [RFC6241] and are not redefined here:

- client
- server
- configuration data
- state data

The following terms are defined in [RFC7950] and are not redefined here:

o augment

o data model

o data node

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

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

1.2. Tree diagram

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

2. Definitions

This document uses the following terms:

Service Provider (SP): The organization (usually a commercial undertaking) responsible for operating the network that offers VPN services to clients and customers.

Customer Edge (CE) Device: Equipment that is dedicated to a particular customer and is directly connected to one or more PE devices via attachment circuits. A CE is usually located at the customer premises, and is usually dedicated to a single VPN, although it may support multiple VPNs if each one has separate attachment circuits. The CE devices can be routers, bridges, switches, or hosts.

Provider Edge (PE) Device: Equipment managed by the SP that can support multiple VPNs for different customers, and is directly connected to one or more CE devices via attachment circuits. A PE is usually located at an SP point of presence (PoP) and is managed by the SP.

Administrative Domain: A collection of End Systems, Intermediate Systems, and subnetworks operated by a single organization or administrative authority. The components which make up the domain are assumed to interoperate with a significant degree of mutual trust among themselves, but interoperate with other Administrative Domains in a mutually suspicious manner [RFC1136].

A group of hosts, routers, and networks operated and managed by a single organization. Routing within an Administrative Domain is based on a consistent technical plan. An Administrative Domain is viewed from the outside, for purposes of routing, as a cohesive entity, of which the internal structure is unimportant. Information passed by other Administrative Domains is trusted less than information from one's own Administrative Domain.

Administrative Domains can be organized into a loose hierarchy that reflects the availability and authoritativeness of routing information. This hierarchy does not imply administrative containment, nor does it imply a strict tree topology.

Routing Domain: A set of End Systems and Intermediate Systems which operate according to the same routing procedures and which is wholly contained within a single Administrative Domain [RFC1136].

A Routing Domain is a set of Intermediate Systems and End Systems bound by a common routing procedure; namely: they are using the same set of routing metrics, they use compatible metric measurement techniques, they use the same information distribution protocol, and they use the same path computation algorithm" An Administrative Domain may contain multiple Routing Domains. A Routing Domain may never span multiple Administrative Domains.

An Administrative Domain may consist of only a single Routing Domain, in which case they are said to be Congruent. A congruent Administrative Domain and Routing Domain is analogous to an Internet Autonomous System.

Access point(AP): Describe an VPN's end point characteristics and its reference to a Termination Point (TP) of the Provider Edge (PE) Node; used as service access point for connectivity service segment in the end-to-end manner and per administrative domain.

Site: Represent a connection of a customer office to one or more VPN services and contain a list of network accesses associated with the site. Each network access can connect to different VPN service.

Segment VPN Describe generic information about a VPN in a single administrative domain, and specific information about APs that connect the Segment VPN to sites or to other Segment VPNs.

Composed VPN Describe generic end-to-end information about a VPN that spans multiple administrative domains, and specific customer-facing information about APsconnecting to each site.

3. Service Model Usage

```
                                     +
          Customer Facing Interface  |  L2SM or L3SM
                              +------v---------------+
                              | Service orchestration |
                              +------+---------------+
                                     | Composed VPN
                              +------v--------------+
                              | Network Orchestrator |
                              +----+---+-------------+
                                   |   |
                         +---------+   +-----------+
                         |                         |
                  +------+--------+          +------+---------+
                  |Config manager1|          | Config manager2|
                  +------+--------+          +------+---------+
                         |                          |
               +---------+-----------+  +----------+----------+
               |      AS1 L2VPN      |  |      AS2  L3VPN     |
  +--------+   | +------+   +------+ |  | +------+   +------+ |  +--------+
  | Site1 +----+ PE11 +---+ PE12 +-------+ PE21 +---+ PE22 +----+ Site2 |
  +--------+   | +------+   +------+ |  | +------+   +------+ |  +--------+
               +---------------------+  +---------------------+
```

Figure 1: Service Model Usage

In the above use case, the network orchestrator controls and manages the two distinct network domains, each controlled or managed by their own management system or domain controller. There are two typical ways to deploy the composed VPN model:

One typical scenario would be to use the model as an independent model. The orchestration layer could use composed VPN model as an input, and translate it to segmented VPN model for each administrative domain. And the domain management system could further configure network elements based on configuration obtained from the segment VPN.

The other scenario is to use customer facing model such as L3SM service model as an input for the service orchestration layer that will be responsible for translating the parameters of VPN and site in L3SM model to the corresponding parameters of the composed VPN model, then with extra provisioning parameters added ,the composed VPN model can be further broken down into per domain segmented VPN model and additional Access point configuration.

The usage of this composed VPN model is not limited to this example; it can be used by any component of the management system but not directly by network elements.

4. The Composed VPN Service Model

A composed VPN represents an end-to-end IP or Ethernet connectivity between the access points of PE where the AP can interconnect with the enterprise customer's network or other types of overlay network. The Composed VPN model provides a common understanding of how the corresponding composed VPN service is to be deployed in an end to end manner over the multi-domain infrastructure.

This document presents the Composed VPN Service Delivery Model using the YANG data modeling language [RFC7950] as a formal language that is both human-readable and parsable by software for use with protocols such as NETCONF [RFC6241] and RESTCONF [RFC8040].

4.1. VPN Service Types

From a technology perspective, a Composed VPN can be classified into three categories based on the domain specific VPN types including L2VPN and L3VPN, see Figure 2. And in each category, the interworking option may vary depending on the inter-domain technology, such as IP or MPLS forwarding. In some cases, the number of transit domain can be zero or multiple.

Composed VPN	Domain 1 (source)	Domain 2 (transit)	..	Domain N (dest)	Interworking Option
L3VPN	L2VPN	L2VPN	..	L3VPN	Option A
L3VPN	L3VPN	L3VPN	..	L3VPN	OptionA/B/C
L2VPN	L2VPN	L2VPN	..	L2VPN	OptionA/B/C

Figure 2: Composed VPN classification

4.2. Composed VPN Physical Network Topology

Figure 3 describes a scenario where connectivity in the form of an L3VPN is provided across a Mobile Backhaul Network. The network has two ASes: connectivity across AS A is achieved with an L2VPN, and across AS B an L3VPN. The ASes are interconnected, and the composed VPN is achieved by interconnecting the L2VPN with the L3VPN.

```
 +---------------------------Composed VPN: L3VPN 1----------------------+
 |                            (AP 1,2,7,8)                              |
 |                                                                      |
 |                                                                      |
 |                                                                      |
 |+---SegVPN:VLL1.1(AP 1,3)---+             +------ SegVPN:L3VPN1.3 ----+
 |           VLL1.2(AP 2,4)   |             |       (AP 5,6,7,8)        |
 |                            |             |                           |
 |                            |             |                           |
 |    ___________________     |             |     ___________________   |
 |   /       AS A         |   |Inter-AS link|   |        AS B           |
 |  |                     |   |             |   |              +------+ |
 |  +------+              |   |             |   |              |PE2    /-\
 /-\|  PE1 |          ++++++++|             |++++++++          |      |AP8|
|AP1|      |          + ASBRA /-\          /-\ ASBRB+          |       \-/
 \-/|      |          +      |AP3  _______|AP5      +          +------+
    +------+          +       \-/          \-/      +                 |
    |                 +       /-\  _______ /-\      +                 |
    |                 +      |AP4         |AP6      +                 |
    +------+          +++++++ \-/          \-/+++++++          +------+
 /-\|      |              |                     |              |       /-\
|AP2|PE4   |              |                     |              |PE3   |AP7|
 \-/|      |              |                     |              |       \-/
    +------+              |                     |              +------+
     \                   /                       \                   /
      ___________________                         ___________________
```

Figure 3: Mobile Backhaul Network Scenario

The Composed VPN is a service that provides connectivity between AP1, AP2, AP7 and AP8. As the APs of the VPN are spanning the two domains, the ASBR A and B and their associated links are required to be identified. Based on the decomposition, two Segment VPN could be constructed to provide per domain connections. Segment VPN 1.1 and Segment VPN 1.2 are connections between AP 1,2,3,4 in the domain A of access metro network, which are L2VPN. Segment VPN 1.3 is the connection between AP 5,6,7,8 in the core network, which is L3VPN. The ASBR A and B at the edge of the access metro network is performing the VPN stitching between Layer 2 VPN and Layer 3 VPN using the technology such as bridging or other interconnection technology.

The operator can predefine several VPN provisioning policies based on the offered business. The policy description may include the naming, path selection, VPN concatenation rules,and resource pools, such as route target, route distinguisher. How VPN provision policies configuration of network elements is done is out of scope of the document.

5. Design of the Data Model

The idea of the composed VPN model is to decompose an end-to-end L2VPN or L3VPN service across multiple administrative domains into point-to-point VPN segments or multi-point VPN segments in each administrative domain, and to stich these segments together by using different interworking options. Therefore, a complete composed VPN instance consists of:

- o One composed VPN with corresponding composed VPN set of parameter
- o Two or more APs, each with a corresponding set of AP parameters
- o One or more segment VPN with corresponding segment VPN set of parameter

Similar to the L3SM [RFC8299] and L2SM [RFC8466] modelling structure, the composed VPN model consists of two main components, the VPN component and the AP component.

The figure below describes the overall structure of the YANG module:

```
module: ietf-composed-vpn-svc
  +--rw composed-vpns
     +--rw composed-vpn* [vpn-id]
        +--rw vpn-id           yang:uuid
        +--rw vpn-name?        string
        +--rw customer-name?   yang:uuid
        +--rw topo?            svpn:vpn-topology
        +--rw service-type?    svpn:service-type
        +--rw tunnel-type?     svpn:tunnel-type
        +--rw admin-state?     svpn:admin-state
        +--ro oper-State?      svpn:oper-state
        +--ro sync-state?      svpn:sync-state
        +--rw start-time?      yang:date-and-time
        +--rw segment-vpn* [vpn-id]
        |  +--rw vpn-id          yang:uuid
        |  +--rw vpn-name?       string
        |  +--rw service-type?   service-type
        |  +--rw topo?           vpn-topology
        |  +--rw tunnel-type?    tunnel-type
        |  +--rw admin-state?    admin-state
        |  +--ro oper-state?     oper-state
        |  +--ro sync-state?     sync-state
```

```
     |  +--rw access-point* [tp-id]
     |     +--rw tp-id                                yang:uuid
     |     +--rw tp-common-attribute
     |     |  +--rw tp-id?                 yang:uuid
     |     |  +--rw tp-name?               string
     |     |  +--rw node-id?               yang:uuid
     |     |  +--rw access-point-type?   access-point-type
     |     |  +--rw inter-as-option?     enumeration
     |     |  +--rw topology-role?       topology-role
     |     +--rw peer-remote-node
     |     |  +--rw remote-id?           yang:uuid
     |     |  +--rw location?            string
     |     |  +--rw remote-tp-address?   inet:ip-address
     |     |  +--rw remote-node-id?      yang:uuid
     |     |  +--rw remote-tp-id?        yang:uuid
     |     +--rw tp-connection-specific-attribute
     |     |  +--rw connection* [connection-class]
     |     |  |  +--rw connection-class    layer-rate
     |     |  |  +--rw (connection-type)?
     |     |  |     +--:(lr-eth)
     |     |  |     |  +--rw eth
     |     |  |     |     +--rw access-type?   eth-encap-type
     |     |  |     |     +--rw (accessVlanValue)?
     |     |  |     |     |  +--:(qinq)
     |     |  |     |     |  |  +--rw qinq
     |     |  |     |     |  |     +--rw cvlan*   uint64
     |     |  |     |     |  |     +--rw svlan?   uint64
     |     |  |     |     |  +--:(dot1q)
     |     |  |     |     |     +--rw dot1q
     |     |  |     |     |        +--rw dot1q*   uint64
     |     |  |     |     +--rw vlan-action?   ethernet-action
     |     |  |     |     +--rw action?        string
     |     |  |     +--:(lr-ip)
     |     |  |     |  +--rw ip
     |     |  |     |     +--rw ip-address?   inet:ip-address
     |     |  |     |     +--rw mtu?          uint64
     |     |  |     +--:(lr-pw)
     |     |  |        +--rw pw
     |     |  |           +--rw control-word?   boolean
     |     |  |           +--rw vlan-action?    pwtagmode
     |     |  +--rw security-attribute
     |     |  |  +--rw security
     |     |  |     +--rw authentication
     |     |  |     +--rw encryption {encryption}?
     |     |  |        +--rw enabled?              boolean
     |     |  |        +--rw layer?                enumeration
     |     |  |        +--rw algorithm?            string
     |     |  |        +--rw (key-type)?
```

```
        |     |  |           +--:(psk)
        |     |  |              +--rw preshared-key?   string
        |     |  +--rw qos-attribute
        |     |  |  +--rw svc-input-bandwidth     uint64
        |     |  |  +--rw svc-output-bandwidth    uint64
        |     |  |  +--rw svc-mtu                 uint16
        |     |  |  +--rw qos {qos}?
        |     |  |     +--rw qos-classification-policy
        |     |  |     |  +--rw rule* [id]
        |     |  |     |     +--rw id                         string
        |     |  |     |     +--rw (match-type)?
        |     |  |     |     |  +--:(match-flow)
        |     |  |     |     |  |  +--rw match-flow
        |     |  |     |     |  |     +--rw dscp?                 inet:dscp
        |     |  |     |     |  |     +--rw exp?                  inet:dscp
        |     |  |     |     |  |     +--rw dot1p?                uint8
        |     |  |     |     |  |     +--rw ipv4-src-prefix?     inet:ipv4-prefi
x
        |     |  |     |     |  |     +--rw ipv6-src-prefix?     inet:ipv6-prefi
x
        |     |  |     |     |  |     +--rw ipv4-dst-prefix?     inet:ipv4-prefi
x
        |     |  |     |     |  |     +--rw ipv6-dst-prefix?     inet:ipv6-prefi
x
        |     |  |     |     |  |     +--rw l4-src-port?         inet:port-numbe
r
        |     |  |     |     |  |     +--rw peer-remote-node*    string
        |     |  |     |     |  |     +--rw l4-src-port-range
        |     |  |     |     |  |     |  +--rw lower-port?   inet:port-number
        |     |  |     |     |  |     |  +--rw upper-port?   inet:port-number
        |     |  |     |     |  |     +--rw l4-dst-port?         inet:port-numbe
r
        |     |  |     |     |  |     +--rw l4-dst-port-range
        |     |  |     |     |  |     |  +--rw lower-port?   inet:port-number
        |     |  |     |     |  |     |  +--rw upper-port?   inet:port-number
        |     |  |     |     |  |     +--rw src-mac?             yang:mac-addres
s
        |     |  |     |     |  |     +--rw dst-mac?             yang:mac-addres
s
        |     |  |     |     |  |     +--rw protocol-field?      union
        |     |  |     |     |  +--:(match-application)
        |     |  |     |     |     +--rw match-application?   identityref
        |     |  |     |     +--rw target-class-id?           string
        |     |  |     +--rw qos-profile
        |     |  |        +--rw (qos-profile)?
        |     |  |           +--:(standard)
        |     |  |           |  +--rw profile?   string
        |     |  |           +--:(custom)
        |     |  |              +--rw classes {qos-custom}?
        |     |  |                 +--rw class* [class-id]
        |     |  |                    +--rw class-id      string
        |     |  |                    +--rw direction?    identityref
        |     |  |                    +--rw rate-limit?   decimal64
        |     |  |                    +--rw latency
        |     |  |                    |  +--rw (flavor)?
        |     |  |                    |     +--:(lowest)
```

```
        |     |  |                    |     |  +--rw use-lowest-latency?   empty
        |     |  |                    |     +--:(boundary)
        |     |  |                    |        +--rw latency-boundary?     uint1
6
        |     |  |                    +--rw jitter
        |     |  |                    |  +--rw (flavor)?
        |     |  |                    |     +--:(lowest)
        |     |  |                    |     |  +--rw use-lowest-jitter?   empty
        |     |  |                    |     +--:(boundary)
        |     |  |                    |        +--rw latency-boundary?    uint32
        |     |  |                    +--rw bandwidth
        |     |  |                       +--rw guaranteed-bw-percent    decimal6
4
        |     |  |                       +--rw end-to-end?              empty
        |     |  +--rw protection-attribute
        |     |     +--rw access-priority?   uint32
        |     +--rw routing-protocol* [type]
        |        +--rw type            protocol-type
        |        +--rw (para)?
        |           +--:(static)
        |           |  +--rw static* [index]
        |           |     +--rw index               uint32
        |           |     +--rw dest-cidr?          string
        |           |     +--rw egress-tp?          yang:uuid
        |           |     +--rw route-preference?   string
        |           |     +--rw next-hop?           inet:ip-address
        |           +--:(bgp)
        |              +--rw bgp* [index]
        |                 +--rw index                uint32
        |                 +--rw autonomous-system    uint32
        |                 +--rw address-family*      address-family
        |                 +--rw max-prefix?          int32
        |                 +--rw peer-address?        inet:ip-address
        |                 +--rw crypto-algorithm     identityref
        |                 +--rw key-string
        |                    +--rw (key-string-style)?
        |                       +--:(keystring)
        |                       |  +--rw keystring?            string
        |                       +--:(hexadecimal) {hex-key-string}?
        |                          +--rw hexadecimal-string?   yang:hex-string
        +--rw access-point* [tp-id]
           +--rw tp-id                                 yang:uuid
           +--rw tp-name?             string
           +--rw node-id?             yang:uuid
           +--rw access-point-type?   access-point-type
           +--rw inter-as-option?     enumeration
           +--rw topology-role?       topology-role
           +--rw peer-remote-node
           |  +--rw remote-id?           yang:uuid
           |  +--rw location?            string
```

```
         |  +--rw remote-tp-address?   inet:ip-address
         |  +--rw remote-node-id?      yang:uuid
         |  +--rw remote-tp-id?        yang:uuid
         +--rw tp-connection-specific-attribute
         |  +--rw connection* [connection-class]
         |  |  +--rw connection-class    layer-rate
         |  |  +--rw (connection-type)?
         |  |     +--:(lr-eth)
         |  |     |  +--rw eth
         |  |     |     +--rw access-type?   eth-encap-type
         |  |     |     +--rw (accessVlanValue)?
         |  |     |     |  +--:(qinq)
         |  |     |     |  |  +--rw qinq
         |  |     |     |  |     +--rw cvlan*   uint64
         |  |     |     |  |     +--rw svlan?   uint64
         |  |     |     |  +--:(dot1q)
         |  |     |     |     +--rw dot1q
         |  |     |     |        +--rw dot1q*   uint64
         |  |     |     +--rw vlan-action?   ethernet-action
         |  |     |     +--rw action?        string
         |  |     +--:(lr-ip)
         |  |     |  +--rw ip
         |  |     |     +--rw ip-address?   inet:ip-address
         |  |     |     +--rw mtu?          uint64
         |  |     +--:(lr-pw)
         |  |        +--rw pw
         |  |           +--rw control-word?   boolean
         |  |           +--rw vlan-action?    pwtagmode
         |  +--rw security-attribute
         |  |  +--rw security
         |  |     +--rw authentication
         |  |     +--rw encryption {encryption}?
         |  |        +--rw enabled?               boolean
         |  |        +--rw layer?                 enumeration
         |  |        +--rw algorithm?             string
         |  |        +--rw (key-type)?
         |  |           +--:(psk)
         |  |              +--rw preshared-key?   string
         |  +--rw qos-attribute
         |  |  +--rw svc-input-bandwidth     uint64
         |  |  +--rw svc-output-bandwidth    uint64
         |  |  +--rw svc-mtu                 uint16
         |  |  +--rw qos {qos}?
         |  |     +--rw qos-classification-policy
         |  |     |  +--rw rule* [id]
         |  |     |     +--rw id                          string
         |  |     |     +--rw (match-type)?
         |  |     |     |  +--:(match-flow)
```

```
   |  |     |     |  |  +--rw match-flow
   |  |     |     |  |     +--rw dscp?                inet:dscp
   |  |     |     |  |     +--rw exp?                 inet:dscp
   |  |     |     |  |     +--rw dot1p?               uint8
   |  |     |     |  |     +--rw ipv4-src-prefix?     inet:ipv4-prefix
   |  |     |     |  |     +--rw ipv6-src-prefix?     inet:ipv6-prefix
   |  |     |     |  |     +--rw ipv4-dst-prefix?     inet:ipv4-prefix
   |  |     |     |  |     +--rw ipv6-dst-prefix?     inet:ipv6-prefix
   |  |     |     |  |     +--rw l4-src-port?         inet:port-number
   |  |     |     |  |     +--rw peer-remote-node*    string
   |  |     |     |  |     +--rw l4-src-port-range
   |  |     |     |  |     |  +--rw lower-port?   inet:port-number
   |  |     |     |  |     |  +--rw upper-port?   inet:port-number
   |  |     |     |  |     +--rw l4-dst-port?         inet:port-number
   |  |     |     |  |     +--rw l4-dst-port-range
   |  |     |     |  |     |  +--rw lower-port?   inet:port-number
   |  |     |     |  |     |  +--rw upper-port?   inet:port-number
   |  |     |     |  |     +--rw src-mac?             yang:mac-address
   |  |     |     |  |     +--rw dst-mac?             yang:mac-address
   |  |     |     |  |     +--rw protocol-field?      union
   |  |     |     |  +--:(match-application)
   |  |     |     |     +--rw match-application?   identityref
   |  |     |     +--rw target-class-id?           string
   |  |     +--rw qos-profile
   |  |        +--rw (qos-profile)?
   |  |           +--:(standard)
   |  |           |  +--rw profile?   string
   |  |           +--:(custom)
   |  |              +--rw classes {qos-custom}?
   |  |                 +--rw class* [class-id]
   |  |                    +--rw class-id      string
   |  |                    +--rw direction?    identityref
   |  |                    +--rw rate-limit?   decimal64
   |  |                    +--rw latency
   |  |                    |  +--rw (flavor)?
   |  |                    |     +--:(lowest)
   |  |                    |     |  +--rw use-lowest-latency?   empty
   |  |                    |     +--:(boundary)
   |  |                    |        +--rw latency-boundary?     uint16
   |  |                    +--rw jitter
   |  |                    |  +--rw (flavor)?
   |  |                    |     +--:(lowest)
   |  |                    |     |  +--rw use-lowest-jitter?   empty
   |  |                    |     +--:(boundary)
   |  |                    |        +--rw latency-boundary?    uint32
   |  |                    +--rw bandwidth
   |  |                       +--rw guaranteed-bw-percent    decimal64
   |  |                       +--rw end-to-end?              empty
```

```
           |  +--rw protection-attribute
           |     +--rw access-priority?   uint32
           +--rw routing-protocol* [type]
              +--rw type            protocol-type
              +--rw (para)?
                 +--:(static)
                 |  +--rw static* [index]
                 |     +--rw index                uint32
                 |     +--rw dest-cidr?           string
                 |     +--rw egress-tp?           yang:uuid
                 |     +--rw route-preference?    string
                 |     +--rw next-hop?            inet:ip-address
                 +--:(bgp)
                    +--rw bgp* [index]
                       +--rw index                 uint32
                       +--rw autonomous-system     uint32
                       +--rw address-family*       address-family
                       +--rw max-prefix?           int32
                       +--rw peer-address?         inet:ip-address
                       +--rw crypto-algorithm      identityref
                       +--rw key-string
                          +--rw (key-string-style)?
                             +--:(keystring)
                             |  +--rw keystring?            string
                             +--:(hexadecimal) {hex-key-string}?
                                +--rw hexadecimal-string?   yang:hex-string
```

5.1. VPN Hierarchy

The composed VPN and segment VPN contain the following common parameters:

o vpn-id: Refers to an internal reference for this VPN service

o vpn-service-type: Combination of L3VPN service type and L2VPN service type per [RFC8466] and [RFC8299], including VPWS,VPLS,EVPN and L3VPN.

o vpn-topology: Combination of L3VPN topology and L2VPN topology, including hub-spoke, any-to-any and point-to-point.

o Tunnel-type:MPLS,MPLS-TP,SR,SRv6

Suppose a composed VPN is a L3VPN which could initially has sites connected to a single SP domain and may later add more sites to other domains in the SP network. Thus, a composed VPN could has one segment VPN at the beginning, and later has more segment VPNs.

5.2. Access Point(AP)

As the site containers of the L3SM and L2SM represent the connection characteristics that the CE connects to the provider network from the perspective of the customer, AP represents the connection characteristics that the PE connects to VPN from the perspective of the provider. Therefore, there are two main aspects relates to the AP modelling:

- o The AP component under composed VPN container describes the intent parameters mapping from the L3SM and L2SM, and the AP component under the segment VPN container describes the configuration parameters of the specific domain derived from the decomposition of composed VPN model.

- o In a specific segment VPN, the AP component not only describes the CE-PE connection, but also defines inter-domain connection parameters between ASBR peer. The connection between PE and ASBR is related to configuration of network elements and not part of segment VPN model.

5.2.1. AP peering with CE

The AP parameters contains the following group of parameters:

Basic AP parameters: topology role could be hub role, leaf role

Connection: has a knob to accommodate either Layer2 or Layer 3 data plane connection

Control plane peering: has a knob to accommodate either Layer 2 protocol or Layer 3 routing protocol

QoS profile and QoS-classification-policy: has a knob to accommodate either Layer 2 QoS profile and qos-classification-policy or Layer 3 QoS profile and Qos-classification-policy, to describe both per AP bandwidth and per flow QoS.

Security Policy: has a knob to accommodate either Layer 2 QoS profile and qos-classification-policy or Layer 3 QoS profile and qos-classification-policy, to describe both per AP bandwidth and per flow QoS.

Although both the composed VPN and segment VPN use the AP to describe the connection parameters of the CE and the PE, the AP parameter of the composed VPN may not be directly mapped to the AP parameters of the segment VPN. For example, a composed VPN is a L3VPN with one of its AP which specifies the IP connection parameters and per flow QOS requirement. During decomposition, depending on the capability of the accessed domain which the segment VPN resides, the AP of the segment VPN could only support Ethernet connection and per port bandwidth guarantee. Therefore, the AP could only configure with L2 connection and per AP bandwidth setting.

5.2.2. AP peering for inter-domains connection

The AP which describes the inter-domains connection could only exist in segment VPN. There are three options in connecting segment VPN across inter-domain link. With L3VPN, L2VPN or mixture, the option could be:

Interworking Option	AP type	AP CP remote peer	AP DP
Option A	ASBR LTP	ASBR	Interface
Option B	ASBR	ASBR	LSP label
Option C	PE,ASBR	remote PE	LSP label

The AP parameters contains the following group of parameters:

Basic AP parameters: Inter-AS interworking option could be Option A, Option B or Option C.

Connection: only specifies in Options A, Option B and C use dynamically allocated MPLS labels.

Control plane peering: BGP peering or static routing.

QoS profile and QoS-classification-policy: only applicable in Options A, Option B and C can only use MPLS EXP to differentiate the traffic.

Security policy: Options A use the similar mechanism like CE-PE peering, Option B could use BGP authentication to secure control plane communication and enable mpls label security, and Option C depends on the trust between the inter-domains.

5.2.2.1. Secure inter-domain connection

This model is applied to a single SP. Although there are different domain separation, implicit trust exists between the ASs because they have the same operational control, for example from orchestrator's perspective.

The model specifies different security parameters depending on the various Inter-AS options:

o Option A uses interfaces or subinterfaces between autonomous system border routers (ASBRs) to keep the VPNs separate, so there is strict separation between VPNs.

o Option B can be secured with configuration on the control plane and the data plane. On the control plane, the session can be secured by use of peer authentication of BGP with message digest 5 (MD5) and TCP Authentication Option(TCP-AO), maximum route limits per peer and per VPN, dampening, and so on. In addition, prefix filters can be deployed to control which routes can be received from the other AS. On the data plane, labeled packets are exchanged. The label is derived from the MP-eBGP session; therefore, the ASBR announcing a VPN-IPv4 prefix controls and assigns the label for each prefix it announces. On the data plane, the incoming label is then checked to verify that this label on the data plane has really been assigned on the control plane. Therefore, it is impossible to introduce fake labels from one AS to another. The Authentication parameter could be set under the BGP peering configuration. An MPLS label security could be enabled under the connection node.

o Option C can also be secured well on the control plane, but the data plane does not provide any mechanism to check and block the packets to be sent into the other AS. On the control plane, model C has two interfaces between autonomous systems: The ASBRs exchange IPv4 routes with labels via eBGP. The purpose is to propagate the PE loopback addresses to the other AS so that LSPs can be established end to end. The other interface is the RRs exchange VPN-IPv4 routes with labels via multihop MP-eBGP. The prefixes exchanged can be controlled through route maps, equally the route targets. On the data plane, the traffic exchanged between the ASBRs contains two labels. One is VPN label set by the ingress PE to identify the VPN. The other is PE label Specifies the LSP to the egress PE. The Authentication and routing policy parameter could be set under the BGP peering configuration.

The security options supported in the model are limited but may be extended via augmentation.

5.2.2.2. Inter-domain QoS decomposition

The APs connected between the domains are aggregation points, and traffic from different CEs of the combined VPN cross-domain will interact through these aggregation points. To provide consistent QoS configuration, when several domains are involved in the provisioning of a VPN, topology, domain functionality and other factors need to be considered.

Option A can achieve most granular QoS implementation since IP traffic passes the inter-domain connection. Thus, Option A can set configuration with per sub-interface and IP DSCP. Option B and Option C only provide MPLS EXP differentiation. QoS mechanisms that are applied only to IP traffic cannot be carried.

In some cases, there is need to re-mark packets at Layer 3 to indicate whether traffic is in agreement. Because MPLS labels include 3 bits that commonly are used for QoS marking, it is possible for "tunnel DiffServ" to preserve Layer 3 DiffServ markings through a service provider's MPLS VPN cloud while still performing re-marking (via MPLS EXP bits) within the cloud to indicate in- or out-of-agreement traffic.

6. Composed VPN YANG Module

```
<CODE BEGINS> file "ietf-composed-vpn-svc.yang"
module ietf-composed-vpn-svc {
    namespace "urn:ietf:params:xml:ns:yang:ietf-composed-vpn-svc" ;
    prefix composed-vpn ;
    import ietf-yang-types {
        prefix yang;
    }
    import ietf-segment-vpn {
        prefix segment-vpn;
    }
    organization "IETF OPSAWG Working Group";
    contact "
     WG Web:   <https://datatracker.ietf.org/wg/opsawg>
        WG List:  <mailto:netmod@ietf.org>

        Editor:   Roni Even
                  <mailto:roni.even@huawei.com>
                  Bo Wu
                  <mailto:lana.wubo@huawei.com>
                  Qin Wu
                  <mailto:bill.wu@huawei.com>
                  Ying Cheng
                  <mailto:chengying10@chinaunicom.cn>";
```

```
    description "ietf-compsed-vpn";
    revision 2018-08-21 {
        reference "draft-evenwu-opsawg-yang-composed-vpn-00";
    }

    grouping vpn-basic {
        description "VPNBasicInfo Grouping.";
        leaf topo {
            type segment-vpn:vpn-topology;
            description "current support for full-mesh and
            point_to_multipoint(hub-spoke), others is reserved for
            future extensions." ;
        }
        leaf service-type {
            type segment-vpn:service-type;
            description "current support for mpls l3vpn/vxlan/L2VPN/hybrid
            VPN overlay, others is reserved for future extensions." ;
        }
        leaf tunnel-type {
            type segment-vpn:tunnel-type;
            description "mpls|vxlan overlay l3vpn|eth over sdh|nop";
        }
        leaf admin-state {
            type segment-vpn:admin-state;
            description "administrative status." ;
         }
       leaf oper-State {
            type segment-vpn:oper-state;
            config false;
            description "Operational status." ;
        }
        leaf sync-state {
            type segment-vpn:sync-state;
            config false;
            description "Sync status." ;
        }
        leaf start-time {
            type yang:date-and-time;
            description "Service lifecycle: request for service start
            time." ;
        }
    }

    container composed-vpns{
        description "";
        list composed-vpn {
            key "vpn-id";
            description "List for composed VPNs.";
```

```
            uses composedvpn;
        }
    }

    grouping composedvpn {
        description "ComposedVPN Grouping.";
        leaf vpn-id {
            type yang:uuid;
            description "Composed VPN identifier." ;
        }
        leaf vpn-name {
            type string  {length "0..200";}
            description "Composed VPN Name. Local administration meaning" ;
        }
        leaf customer-name {
            type yang:uuid;
            description
              "Name of the customer that actually uses the VPN service.
              In the case that any intermediary (e.g., Tier-2 provider
              or partner) sells the VPN service to their end user
              on behalf of the original service provider (e.g., Tier-1
              provider), the original service provider may require the
              customer name to provide smooth activation/commissioning
              and operation for the service." ;
        }
        uses vpn-basic;
        list segment-vpn {
            key "vpn-id";
            description "SegVpn list ";
            uses segment-vpn:VPN;
        }
        list access-point {
            key "tp-id";
            description "TP list of the access links which associated
            with CE and PE";
            uses segment-vpn:pe-termination-point;
        }
    }
}
<CODE ENDS>
```

7. Segment VPN YANG Module

```
<CODE BEGINS> file "ietf-segment-vpn.yang"
module ietf-segment-vpn {
  namespace "urn:ietf:params:xml:ns:yang:ietf-segment-vpn";
  prefix segment-vpn;
```

```
  import ietf-yang-types {
    prefix yang;
  }
  import ietf-inet-types {
    prefix inet;
  }
  import ietf-key-chain {
    prefix keychain;
  }
  import ietf-netconf-acm {
    prefix nacm;
  }

  organization
    "IETF OPSAWG Working Group";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/opsawg>
     WG List:  <mailto:netmod@ietf.org>

     Editor:
       Roni Even
          <mailto:roni.even@huawei.com>
          Bo Wu
          <mailto:lana.wubo@huawei.com>
          Qin Wu
          <mailto:bill.wu@huawei.com>
          Cheng Ying
          <mailto:chengying10@chinaunicom.cn>";
  description
    "This YANG module defines a generic service configuration
     model for segment VPNs.";

  revision 2019-01-30 {
    reference
      "draft-opsawg-evenwu-yang-composed-vpn-02";
  }

  feature encryption {
    description
      "Enables support of encryption.";
  }

  feature qos {
    description
      "Enables support of classes of services.";
  }

  feature qos-custom {
```

```
    description
      "Enables support of the custom QoS profile.";
  }

  feature hex-key-string {
    description
      "Support hexadecimal key string.";
  }

  identity protocol-type {
    description
      "Base identity for protocol field type.";
  }

  identity tcp {
    base protocol-type;
    description
      "TCP protocol type.";
  }

  identity udp {
    base protocol-type;
    description
      "UDP protocol type.";
  }

  identity icmp {
    base protocol-type;
    description
      "ICMP protocol type.";
  }

  identity icmp6 {
    base protocol-type;
    description
      "ICMPv6 protocol type.";
  }

  identity gre {
    base protocol-type;
    description
      "GRE protocol type.";
  }

  identity ipip {
    base protocol-type;
    description
      "IP-in-IP protocol type.";
```

```
  }

  identity hop-by-hop {
    base protocol-type;
    description
      "Hop-by-Hop IPv6 header type.";
  }

  identity routing {
    base protocol-type;
    description
      "Routing IPv6 header type.";
  }

  identity esp {
    base protocol-type;
    description
      "ESP header type.";
  }

  identity ah {
    base protocol-type;
    description
      "AH header type.";
  }

  identity customer-application {
    description
      "Base identity for customer application.";
  }

  identity web {
    base customer-application;
    description
      "Identity for Web application (e.g., HTTP, HTTPS).";
  }

  identity mail {
    base customer-application;
    description
      "Identity for mail application.";
  }

  identity file-transfer {
    base customer-application;
    description
      "Identity for file transfer application (e.g., FTP, SFTP).";
  }
```

```
  identity database {
    base customer-application;
    description
      "Identity for database application.";
  }

  identity social {
    base customer-application;
    description
      "Identity for social-network application.";
  }

  identity games {
    base customer-application;
    description
      "Identity for gaming application.";
  }

  identity p2p {
    base customer-application;
    description
      "Identity for peer-to-peer application.";
  }

  identity network-management {
    base customer-application;
    description
      "Identity for management application
       (e.g., Telnet, syslog, SNMP).";
  }

  identity voice {
    base customer-application;
    description
      "Identity for voice application.";
  }

  identity video {
    base customer-application;
    description
      "Identity for video conference application.";
  }

  identity qos-profile-direction {
    description
      "Base identity for QoS profile direction.";
  }
```

```
  identity outbound {
    base qos-profile-direction;
    description
      "Identity for outbound direction.";
  }

  identity inbound {
    base qos-profile-direction;
    description
      "Identity for inbound direction.";
  }

  identity both {
    base qos-profile-direction;
    description
      "Identity for both inbound direction
       and outbound direction.";
  }

  typedef access-point-type {
    type enumeration {
      enum ce-peering {
        description
          "indicates access type with connection to CE";
      }
      enum remote-as-peering {
        description
          "indicates access type with connection to ASBR with opion A,B,C ";
      }
    }
    description
      "The access-point-type could be peering with CE or ASBR
       depending on which network that a PE interconnects with.";
  }

  typedef bgp-password-type {
    type string;
    description
      "Authentication Type (None, Simple Password, Keyed MD5,
       Meticulous Keyed MD5, Keyed SHA1, Meticulous Keyed SHA1";
  }

  typedef topology-role {
    type enumeration {
      enum hub {
        description
          "hub";
      }
```

```
      enum spoke {
        description
          "spoke";
      }
      enum other {
        description
          "other";
      }
    }
    description
      "Topo Node Role.";
  }

  typedef qos-config-type {
    type enumeration {
      enum template {
        description
          "standard.";
      }
      enum customer {
        description
          "custom.";
      }
    }
    description
      "Qos Config Type.";
  }

  typedef address-family {
    type enumeration {
      enum ipv4 {
        description
          "IPv4 address family.";
      }
      enum ipv6 {
        description
          "IPv6 address family.";
      }
    }
    description
      "Defines a type for the address family.";
  }

  typedef tp-type {
    type enumeration {
      enum phys-tp {
        description
          "Physical termination point";
```

```
      }
      enum ctp {
        description
          "CTP";
      }
      enum trunk {
        description
          "TRUNK";
      }
      enum loopback {
        description
          "LoopBack";
      }
      enum tppool {
        description
          "TPPool";
      }
    }
    description
      "Tp Type.";
  }

  typedef layer-rate {
    type enumeration {
      enum lr-unknow {
        description
          "Layer Rate UNKNOW.";
      }
      enum lr-ip {
        description
          "Layer Rate IP.";
      }
      enum lr-eth {
        description
          "Layer Rate Ethernet.";
      }
      enum lr_vxlan {
        description
          "Layer Rate VXLAN.";
      }
    }
    description
      "Layer Rate.";
  }

  typedef admin-state {
    type enumeration {
      enum active {
```

```
        description
          "Active status";
      }
      enum inactive {
        description
          "Inactive status";
      }
      enum partial {
        description
          "Partial status";
      }
    }
    description
      "Admin State.";
  }

  typedef oper-state {
    type enumeration {
      enum up {
        description
          "Up status";
      }
      enum down {
        description
          "Down status";
      }
      enum degrade {
        description
          "Degrade status";
      }
    }
    description
      "Operational Status.";
  }

  typedef sync-state {
    type enumeration {
      enum sync {
        description
          "Sync status";
      }
      enum out-sync {
        description
          "Out sync status";
      }
    }
    description
      "Sync Status";
```

```
  }

  typedef eth-encap-type {
    type enumeration {
      enum default {
        description
          "DEFAULT";
      }
      enum dot1q {
        description
          "DOT1Q";
      }
      enum qinq {
        description
          "QINQ";
      }
      enum untag {
        description
          "UNTAG";
      }
    }
    description
      "Ethernet Encap Type.";
  }

  typedef protocol-type {
    type enumeration {
      enum static {
        description
          "Static Routing";
      }
      enum bgp {
        description
          "bgp";
      }
      enum rip {
        description
          "rip";
      }
      enum ospf {
        description
          "ospf";
      }
      enum isis {
        description
          "isis";
      }
    }
```

```
    description
      "Routing Protocol Type";
  }

  typedef tunnel-type {
    type enumeration {
      enum MPLS {
        description
          "MPLS";
      }
      enum MPLS-TP {
        description
          "MPLS-TP";
      }
      enum MPLS-SR {
        description
          "MPLS Segment Routing";
      }
      enum SRv6 {
        description
          "SRv6";
      }
    }
    description
      "VPN Tunnel Type.";
  }

  typedef service-type {
    type enumeration {
      enum l3vpn {
        description
          "l3vpn";
      }
      enum l2vpn {
        description
          "l2vpn";
      }
    }
    description
      "VPN Service Type.";
  }

  typedef vpn-topology {
    type enumeration {
      enum point-to-point {
        description
          "point to point";
      }
```

```
      enum any-to-any {
        description
          "any to any";
      }
      enum hub-spoke {
        description
          "hub and spoke VPN topology.";
      }
      enum hub-spoke-disjoint {
        description
          "Hub and spoke VPN topology where
           Hubs cannot communicate with each other ";
      }
    }
    description
      "Topology.";
  }

  typedef ethernet-action {
    type enumeration {
      enum nop {
        description
          "nop";
      }
      enum untag {
        description
          "UNTAG";
      }
      enum stacking {
        description
          "STACKING";
      }
    }
    description
      "Ethernet Action.";
  }

  typedef color-type {
    type enumeration {
      enum green {
        description
          "green";
      }
      enum yellow {
        description
          "yellow";
      }
      enum red {
```

```
        description
          "red";
      }
      enum all {
        description
          "all";
      }
    }
    description
      "Color Type.";
  }

  typedef action-type {
    type enumeration {
      enum nop {
        description
          "nop";
      }
      enum bandwidth {
        description
          "bandwidth";
      }
      enum pass {
        description
          "pass";
      }
      enum discard {
        description
          "discard";
      }
      enum remark {
        description
          "remark";
      }
      enum redirect {
        description
          "redirect";
      }
      enum recolor {
        description
          "recolor";
      }
      enum addRt {
        description
          "addRt";
      }
    }
    description
```

```
      "Action Type";
  }

  typedef pwtagmode {
    type enumeration {
      enum raw {
        description
          "RAW";
      }
      enum tagged {
        description
          "TAGGED";
      }
    }
    description
      "PWTagMode";
  }

  grouping QinQVlan {
    description
      "QinQVlan Grouping.";
    leaf-list cvlan {
      type uint64;
      description
        "cvlan List.";
    }
    leaf svlan {
      type uint64;
      description
        "svlan.";
    }
  }

  grouping Dot1QVlan {
    description
      "Dot1QVlan Grouping.";
    leaf-list dot1q {
      type uint64;
      description
        "dot1q Vlan List";
    }
  }

  grouping tp-connection-type {
    description
      "Tp Type Spec Grouping.";
    choice connection-type {
      description
```

```
          "Spec Value";
        case lr-eth {
          container eth {
            description
              "ethernetSpec";
            uses ethernet-spec;
          }
        }
        case lr-ip {
          container ip {
            description
              "ipSpec";
            uses ipspec;
          }
        }
        case lr-pw {
          container pw {
            description
              "PwSpec";
            uses pwspec;
          }
        }
      }
    }

    grouping security-authentication {
      container authentication {
        description
          "Authentication parameters.";
      }
      description
        "This grouping defines authentication parameters for a site.";
    }

    grouping security-encryption {
      container encryption {
        if-feature "encryption";
        leaf enabled {
          type boolean;
          default "false";
          description
            "If true, traffic encryption on the connection is required.";
        }
        leaf layer {
          when "../enabled = 'true'" {
            description
              " Require a value for layer when enabled is true.";
          }
```

```
        type enumeration {
          enum layer2 {
            description
              "Encryption will occur at Layer 2.";
          }
          enum layer3 {
            description
              "Encryption will occur at Layer 3.
               For example, IPsec may be used when
               a customer requests Layer 3 encryption.";
          }
        }
        description
          "Layer on which encryption is applied.";
      }
      leaf algorithm {
        type string;
        description
          "Encryption algorithm to be used.";
      }
      choice key-type {
        default "psk";
        case psk {
          leaf preshared-key {
            type string;
            description
              " Pre-Shared Key(PSK) coming from customer.";
          }
        }
        description
          "Type of keys to be used.";
      }
      description
        "Encryption parameters.";
    }
    description
      "This grouping defines encryption parameters for a site.";
  }

  grouping security-attribute {
    container security {
      uses security-authentication;
      uses security-encryption;
      description
        "Site-specific security parameters.";
    }
    description
      "Grouping for security parameters.";
```

```
  }

  grouping flow-definition {
    container match-flow {
      leaf dscp {
        type inet:dscp;
        description
          "DSCP value.";
      }
      leaf exp {
        type inet:dscp;
        description
          "EXP value.";
      }
      leaf dot1p {
        type uint8 {
          range "0..7";
        }
        description
          "802.1p matching.";
      }
      leaf ipv4-src-prefix {
        type inet:ipv4-prefix;
        description
          "Match on IPv4 src address.";
      }
      leaf ipv6-src-prefix {
        type inet:ipv6-prefix;
        description
          "Match on IPv6 src address.";
      }
      leaf ipv4-dst-prefix {
        type inet:ipv4-prefix;
        description
          "Match on IPv4 dst address.";
      }
      leaf ipv6-dst-prefix {
        type inet:ipv6-prefix;
        description
          "Match on IPv6 dst address.";
      }
      leaf l4-src-port {
        type inet:port-number;
        must 'current() < ../l4-src-port-range/lower-port or current() > ../l4-s
rc-port-range/upper-port' {
          description
            "If l4-src-port and l4-src-port-range/lower-port and
             upper-port are set at the same time, l4-src-port
             should not overlap with l4-src-port-range.";
```

```
        }
        description
          "Match on Layer 4 src port.";
      }
      leaf-list peer-remote-node {
        type string;
        description
          "Identify a peer remote node as traffic destination.";
      }
      container l4-src-port-range {
        leaf lower-port {
          type inet:port-number;
          description
            "Lower boundary for port.";
        }
        leaf upper-port {
          type inet:port-number;
          must '. >= ../lower-port' {
            description
              "Upper boundary for port.  If it
               exists, the upper boundary must be
               higher than the lower boundary.";
          }
          description
            "Upper boundary for port.";
        }
        description
          "Match on Layer 4 src port range.  When
           only the lower-port is present, it represents
           a single port.  When both the lower-port and
           upper-port are specified, it implies
           a range inclusive of both values.";
      }
      leaf l4-dst-port {
        type inet:port-number;
        must 'current() < ../l4-dst-port-range/lower-port or current() > ../l4-d
st-port-range/upper-port' {
          description
            "If l4-dst-port and l4-dst-port-range/lower-port
             and upper-port are set at the same time,
             l4-dst-port should not overlap with
             l4-src-port-range.";
        }
        description
          "Match on Layer 4 dst port.";
      }
      container l4-dst-port-range {
        leaf lower-port {
          type inet:port-number;
```

```
          description
            "Lower boundary for port.";
        }
        leaf upper-port {
          type inet:port-number;
          must '. >= ../lower-port' {
            description
              "Upper boundary must be
               higher than lower boundary.";
          }
          description
            "Upper boundary for port.  If it exists,
             upper boundary must be higher than lower
             boundary.";
        }
        description
          "Match on Layer 4 dst port range.  When only
           lower-port is present, it represents a single
           port.  When both lower-port and upper-port are
           specified, it implies a range inclusive of both
           values.";
      }
      leaf src-mac {
        type yang:mac-address;
        description
          "Source MAC.";
      }
      leaf dst-mac {
        type yang:mac-address;
        description
          "Destination MAC.";
      }
      leaf protocol-field {
        type union {
          type uint8;
          type identityref {
            base protocol-type;
          }
        }
        description
          "Match on IPv4 protocol or IPv6 Next Header field.";
      }
      description
        "Describes flow-matching criteria.";
    }
    description
      "Flow definition based on criteria.";
  }
```

```
  grouping service-qos-profile {
    container qos {
      if-feature "qos";
      container qos-classification-policy {
        list rule {
          key "id";
          ordered-by user;
          leaf id {
            type string;
            description
              "A description identifying the
               qos-classification-policy rule.";
          }
          choice match-type {
            default "match-flow";
            case match-flow {
              uses flow-definition;
            }
            case match-application {
              leaf match-application {
                type identityref {
                  base customer-application;
                }
                description
                  "Defines the application to match.";
              }
            }
            description
              "Choice for classification.";
          }
          leaf target-class-id {
            type string;
            description
              "Identification of the class of service.
               This identifier is internal to the administration.";
          }
          description
            "List of marking rules.";
        }
        description
          "Configuration of the traffic classification policy.";
      }
      container qos-profile {
        choice qos-profile {
          description
            "Choice for QoS profile.
             Can be standard profile or customized profile.";
          case standard {
```

```
            description
              "Standard QoS profile.";
            leaf profile {
              type string;
              description
                "QoS profile to be used.";
            }
          }
          case custom {
            description
              "Customized QoS profile.";
            container classes {
              if-feature "qos-custom";
              list class {
                key "class-id";
                leaf class-id {
                  type string;
                  description
                    "Identification of the class of service.
                     This identifier is internal to the
                     administration.";
                }
                leaf direction {
                  type identityref {
                    base qos-profile-direction;
                  }
                  default "both";
                  description
                    "The direction to which the QoS profile
                     is applied.";
                }
                leaf rate-limit {
                  type decimal64 {
                    fraction-digits 5;
                    range "0..100";
                  }
                  units "percent";
                  description
                    "To be used if the class must be rate-limited.
                     Expressed as percentage of the service
                     bandwidth.";
                }
                container latency {
                  choice flavor {
                    case lowest {
                      leaf use-lowest-latency {
                        type empty;
                        description
```

```
                       "The traffic class should use the path with the
                        lowest latency.";
                   }
                 }
                 case boundary {
                   leaf latency-boundary {
                     type uint16;
                     units "msec";
                     default "400";
                     description
                       "The traffic class should use a path with a
                        defined maximum latency.";
                   }
                 }
                 description
                   "Latency constraint on the traffic class.";
               }
               description
                 "Latency constraint on the traffic class.";
             }
             container jitter {
               choice flavor {
                 case lowest {
                   leaf use-lowest-jitter {
                     type empty;
                     description
                       "The traffic class should use the path with the
                        lowest jitter.";
                   }
                 }
                 case boundary {
                   leaf latency-boundary {
                     type uint32;
                     units "usec";
                     default "40000";
                     description
                       "The traffic class should use a path with a
                        defined maximum jitter.";
                   }
                 }
                 description
                   "Jitter constraint on the traffic class.";
               }
               description
                 "Jitter constraint on the traffic class.";
             }
             container bandwidth {
               leaf guaranteed-bw-percent {
```

```
                    type decimal64 {
                      fraction-digits 5;
                      range "0..100";
                    }
                    units "percent";
                    mandatory true;
                    description
                      "To be used to define the guaranteed bandwidth
                       as a percentage of the available service bandwidth.";
                  }
                  leaf end-to-end {
                    type empty;
                    description
                      "Used if the bandwidth reservation
                       must be done on the MPLS network too.";
                  }
                  description
                    "Bandwidth constraint on the traffic class.";
                }
                description
                  "List of classes of services.";
              }
              description
                "Container for list of classes of services.";
            }
          }
        }
        description
          "QoS profile configuration.";
      }
      description
        "QoS configuration.";
    }
    description
      "This grouping defines QoS parameters for a segment network.";
  }

  grouping remote-peer-tp {
    description
      "remote-peer-tp Grouping.";
    leaf remote-id {
      type yang:uuid;
      description
        "Router ID of the remote peer";
    }
    leaf location {
      type string {
        length "0..400";
```

```
      }
      description
        "CE device location ";
    }
    leaf remote-tp-address {
      type inet:ip-address;
      description
        "TP IP address";
    }
    leaf remote-node-id {
      type yang:uuid;
      description
        "directly connected NE node ID, only valid in
         asbr ";
    }
    leaf remote-tp-id {
      type yang:uuid;
      description
        "Directly connected TP id, only valid in asbr";
    }
  }

  grouping tp-connection-specific-attribute {
    description
      "tp connectin specific attributes";
    list connection {
      key "connection-class";
      leaf connection-class {
        type layer-rate;
        description
          "connection class and has one to one
         relation with the corresponding layer.";
      }
      uses tp-connection-type;
      description
        "typeSpecList";
    }
    container security-attribute {
      description
        "tp security Parameters.";
      uses security-attribute;
    }
    container qos-attribute {
      description
        "tp Qos Parameters.";
      uses segment-service-basic;
      uses service-qos-profile;
    }
```

```
    container protection-attribute {
      description
        "tp protection parameters.";
      leaf access-priority {
        type uint32;
        default "100";
        description
          "Defines the priority for the access.
           The higher the access-priority value,
           the higher the preference of the
           access will be.";
      }
    }
  }

  grouping tp-common-attribute {
    description
      "tp-common-attribute Grouping.";
    leaf tp-id {
      type yang:uuid;
      description
        "An identifier for termination point on a node.";
    }
    leaf tp-name {
      type string {
        length "0..200";
      }
      description
        "The termination point Name on a node. It conforms to
         name rule defined in system. Example FE0/0/1, GE1/2/1.1,
         Eth-Trunk1.1, etc";
    }
    leaf node-id {
      type yang:uuid;
      description
        "Identifier for a node.";
    }
    leaf access-point-type {
      type access-point-type;
      description
        "access-point-type, for example:peering with CE ";
    }
    leaf inter-as-option {
      type enumeration {
        enum optiona {
          description
            "Inter-AS Option A";
        }
```

```
        enum optionb {
          description
            "Inter-AS Option B";
        }
        enum optionc {
          description
            "Inter-AS Option C";
        }
      }
      description
        "Foo";
    }
    leaf topology-role {
      type topology-role;
      description
        "hub/spoke role, etc";
    }
  }

  grouping routing-protcol {
    description
      "Routing Protocol Grouping.";
    leaf type {
      type protocol-type;
      description
        "Protocol type";
    }
    choice para {
      description
        "para";
      case static {
        list static {
          key "index";
          uses static-config;
          description
            "staticRouteItems";
        }
      }
      case bgp {
        list bgp {
          key "index";
          uses bgp-config;
          description
            "bgpProtocols";
        }
      }
    }
  }
```

```
  grouping bgp-config {
    description
      "BGP Protocol Grouping.";
    leaf index {
      type uint32;
      description
        "index of BGP protocol item";
    }
    leaf autonomous-system {
      type uint32;
      mandatory true;
      description
        "Peer AS number in case the peer
         requests BGP routing.";
    }
    leaf-list address-family {
      type address-family;
      min-elements 1;
      description
        "If BGP is used on this site, this node
         contains configured value. This node
         contains at least one address family
         to be activated.";
    }
    leaf max-prefix {
      type int32;
      description
        "maximum number limit of prefixes.";
    }
    leaf peer-address {
      type inet:ip-address;
      description
        "peerIp";
    }
    leaf crypto-algorithm {
      type identityref {
        base keychain:crypto-algorithm;
      }
      mandatory true;
      description
        "Cryptographic algorithm associated with key.";
    }
    container key-string {
      description
        "The key string.";
      nacm:default-deny-all;
      choice key-string-style {
        description
```

```
          "Key string styles";
        case keystring {
          leaf keystring {
            type string;
            description
              "Key string in ASCII format.";
          }
        }
        case hexadecimal {
          if-feature "hex-key-string";
          leaf hexadecimal-string {
            type yang:hex-string;
            description
              "Key in hexadecimal string format.  When compared
               to ASCII, specification in hexadecimal affords
               greater key entropy with the same number of
               internal key-string octets.  Additionally, it
               discourages usage of well-known words or
               numbers.";
          }
        }
      }
    }
  }

  grouping static-config {
    description
      "StaticRouteItem Grouping.";
    leaf index {
      type uint32;
      description
        "static item index";
    }
    leaf dest-cidr {
      type string;
      description
        "address prefix specifying the set of
         destination addresses for which the route may be
         used. ";
    }
    leaf egress-tp {
      type yang:uuid;
      description
        "egress tp";
    }
    leaf route-preference {
      type string;
      description
```

```
          "route priority. Ordinary, work route have
           higher priority.";
      }
      leaf next-hop {
        type inet:ip-address;
        description
          "Determines the outgoing interface and/or
           next-hop address(es), or a special operation to be
           performed on a packet..";
      }
    }

    grouping ethernet-spec {
      description
        "Ethernet Spec Grouping.";
      leaf access-type {
        type eth-encap-type;
        description
          "access frame type";
      }
      choice accessVlanValue {
        description
          "accessVlanValue";
        case qinq {
          container qinq {
            description
              "qinqVlan";
            uses QinQVlan;
          }
        }
        case dot1q {
          container dot1q {
            description
              "dot1q";
            uses Dot1QVlan;
          }
        }
      }
      leaf vlan-action {
        type ethernet-action;
        description
          "specify the action when the vlan is matched";
      }
      leaf action {
        type string {
          length "0..100";
        }
        description
```

```
        "specify the action value.";
    }
  }

  grouping pwspec {
    description
      "PwSpec Grouping.";
    leaf control-word {
      type boolean;
      default "false";
      description
        "control Word.";
    }
    leaf vlan-action {
      type pwtagmode;
      description
        "pw Vlan Action.";
    }
  }

  grouping ipspec {
    description
      "IpSpec Grouping.";
    leaf ip-address {
      type inet:ip-address;
      description
        "master IP address";
    }
    leaf mtu {
      type uint64;
      description
        "mtu for ip layer,scope:46~9600";
    }
  }

  grouping VPN {
    description
      "VPN Grouping.";
    leaf vpn-id {
      type yang:uuid;
      description
        "VPN Identifier.";
    }
    leaf vpn-name {
      type string {
        length "0..200";
      }
      description
```

```
         "Human-readable name for the VPN service.";
     }
     leaf service-type {
       type service-type;
       description
         "The service type combines service types from
          RFC8299 (L3SM) and RFC8466 (L2SM),for example L3VPN,VPWS etc.
          It could be augmentated for future extensions.";
     }
     leaf topo {
       type vpn-topology;
       description
         "The VPN topology could be full-mesh,point-to-point
          and hub-spoke, others is reserved for future extensions.";
     }
     leaf tunnel-type {
       type tunnel-type;
       description
         "Tunnel Type:LDP&#65306;LDP Tunnel,RSVP-TE&#65306;RSVP-TE Tunnel
          SR-TE&#65306;SR-TE Tunnel,MPLS-TP&#65306;MPLS-TP Tunnel,VXLAN&#65306;VX
LAN Tunnel
         ";
     }
     leaf admin-state {
       type admin-state;
       description
         "administrative status.";
     }
     leaf oper-state {
       type oper-state;
       config false;
       description
         "Operational status.";
     }
     leaf sync-state {
       type sync-state;
       config false;
       description
         "Sync status.";
     }
     list access-point {
       key "tp-id";
       description
         "TP list of the access links which associated
          with PE and CE or ASBR";
       uses pe-termination-point;
     }
   }
```

```
  grouping pe-termination-point {
    description
      "grouping for termination points.";
    uses tp-common-attribute;
    container peer-remote-node {
      description
        "TP Peering Information, including CE
         peering and ASBR peering.";
    uses remote-peer-tp;
    }
    container tp-connection-specific-attribute {
      description
        "Termination point basic info.";
      uses tp-connection-specific-attribute;
    }
    list routing-protocol {
      key "type";
      description
        "route protocol spec.";
      uses routing-protcol;
    }
  }

  grouping segment-service-basic {
    leaf svc-input-bandwidth {
      type uint64;
      units "bps";
      mandatory true;
      description
        "From the customer site's perspective, the service
         input bandwidth of the connection or download
         bandwidth from the SP to the site.";
    }
    leaf svc-output-bandwidth {
      type uint64;
      units "bps";
      mandatory true;
      description
        "From the customer site's perspective, the service
         output bandwidth of the connection or upload
         bandwidth from the site to the SP.";
    }
    leaf svc-mtu {
      type uint16;
      units "bytes";
      mandatory true;
      description
        "MTU at service level.  If the service is IP,
```

```
         it refers to the IP MTU.  If CsC is enabled,
         the requested 'svc-mtu' leaf will refer to the
         MPLS MTU and not to the IP MTU.";
    }
    description
      "Defines basic service parameters for a site.";
  }

  container segment-vpns {
    list segment-vpn {
      key "index";
      description
        "Segment Vpn list.";
      leaf index {
        type uint32;
        description
          "index of segment VPN in a composed VPN.";
      }
      uses VPN;
    }
    description
      "Container for Segment VPN.";
  }
}
<CODE ENDS>
```

8. Service Model Usage Example

This section provides an example of how a management system can use this model to configure an IP VPN service on network elements.

```
+------------------------------------------------------------------------+
|                                          ------- PE2----- Spoke_Site1 |
|                                          |                             |
| Hub_Site1-----PE1------ASBR1-------- ASBR2                           |
|                                          |                             |
|                                          ---------PE3 ---- Spoke_Site2 |
+----------------|----------|--------------|--------|--------------------+
                 |          |              |        |
                 |<SegVPN1>
 | Inter-AS link|<SegVPN2>
                 |          |              |        |
                 |          |              |        |
                 | Intra-AS |   Inter-AS   |Intra-AS|
                 |                                  |
                 |<--------Composed VPN ----------->|
```

Composed VPN Service Model Usage Example

In this example, we want to achieve the provisioning of an end to end VPN service for three sites using a Hub-and-Spoke VPN service topology. The end to end VPN service is stitched by two segmented VPN.

The following XML snippet describes the overall simplified service configuration of this composed VPN.


```
<?xml version="1.0"?>
<composed-vpns xmlns="urn:ietf:params:xml:ns:yang:ietf-composed-vpn-svc">
     <composed-vpn>
      <vpn-id>12456487</vpn-id>
      <topo>hub-spoke</topo>
      <service-type>hybrid</service-type>
      <segment-vpn>
        <index>1</index>
          <vpn-id>111<vpn-id>
          <topo>hub-spoke</topo>
          <service-type>l2vpn</service-type>
          <access-point>
             <tp-id>ap1-tp1</tp-id>
             <node-id>PE1</node-id>
             <topology-role>hub</topology-role>
             <peer-remote-node>
               <remote-node-id>Hub_Site1</remote-node-id>
             </peer-remote-node>
              <tp-connection-specific-attribute>
               <qos-attribute>
               <svc-mtu>1514</svc-mtu>
               <svc-input-bandwidth>10000000</svc-input-bandwidth>
               <svc-output-bandwidth>10000000</svc-output-bandwidth>
                </qos-attribute>
             </tp-connection-specific-attribute>
             <routing-protocol>
               <type>bgp</type>
               <bgp>
               <as-no>AS1</as-no>
               </bgp>
            </routing-protocol>
            </access-point>
            <access-point>
             <tp-id>ap1-tp2</tp-id>
             <node-id>ASBR1</node-id>
             <topo-role>hub</topo-role>
             <peer-remote-node>
               <remote-node-id>ASBR2</remote-node-id>
             </peer-remote-node>
             <inter-AS-option>Option A</inter-AS-option>
```

```
                    <tp-connection-specific-attribute>
                     <qos-attribute>
                     <svc-mtu>1514</svc-mtu>
                     <svc-input-bandwidth>10000000</svc-input-bandwidth>
                     <svc-output-bandwidth>10000000</svc-output-bandwidth>
                      </qos-attribute>
                   </tp-connection-specific-attribute>
                   <routing-protocol>
                     <type>bgp</type>
                     <bgp>
                     <as-no>AS1</as-no>
                     </bgp>
                  </routing-protocol>
                </access-point>
            </segment-vpn>
            <segment-vpn>
              <index>2</index>
                <vpn-id>222<vpn-id>
                <topo>hub-spoke</topo>
                <service-type>l3vpn</service-type>
                <access-point>
                    <tp-id>ap2-tp2</tp-id>
                    <node-id>PE2</node-id>
                    <topo-role>spoke</topo-role>
                   <peer-remote-node>
                     <remote-node-id>Spoke_Site1</remote-node-id>
                   </peer-remote-node>
                     <qos-attribute>
                     <svc-mtu>1514</svc-mtu>
                     <svc-input-bandwidth>10000000</svc-input-bandwidth>
                     <svc-output-bandwidth>10000000</svc-output-bandwidth>
                      </qos-attribute>
                     <routing-protocol>
                      <type>bgp</type>
                       <bgp>
                       <as-no>ASXXX</as-no>
                       </bgp>
                     </routing-protocol>
                   </access-point>
                   <access-point>
                   <tp-id>ap2-tp1</tp-id>
                   <node-id>PE3</node-id>
                    <topo-role>spoke</topo-role>
                    <peer-remote-node>
                     <remote-node-id>Spoke_Site2</remote-node-id>
                   </peer-remote-node>
                     <qos-attribute>
                     <svc-mtu>1514</svc-mtu>
```

```
                     <svc-input-bandwidth>10000000</svc-input-bandwidth>
                     <svc-output-bandwidth>10000000</svc-output-bandwidth>
                      </qos-attribute>
                   <routing-protocol>
                     <type>bgp</type>
                     <bgp>
                     <as-no>ASXXX</as-no>
                     </bgp>
                  <routing-protocol>
                   </access-point>
                   <access-point>
                   <tp-id>ap2-tp3</tp-id>
                   <node-id>ASBR2</node-id>
                    <topo-role>hub</topo-role>
                   <peer-remote-node>
                     <remote-node-id>ASBR1</remote-node-id>
                   </peer-remote-node>
                    <qos-attribute>
                     <svc-mtu>1514</svc-mtu>
                     <svc-input-bandwidth>10000000</svc-input-bandwidth>
                     <svc-output-bandwidth>10000000</svc-output-bandwidth>
                      </qos-attribute>
                   <routing-protocol>
                     <type>bgp</type>
                     <bgp>
                     <as-no>interAS-1</as-no>
                     </bgp>
                  <routing-protocol>
                </access-point>
            </segment-vpn>
          </composed-vpn>
      </composed-vpns>
```


9. Interaction with other YANG models

As expressed in Section 4, this composed VPN service model is intended to be instantiated in a management system and not directly on network elements.

The management system's role will be to configure the network elements. The management system may be modular and distinguish the component instantiating the service model (let's call it "service component") from the component responsible for network element configuration (let's call it "configuration component"). The service is built from a combination of network elements and protocols configuration which also include various aspects of the underlying network infrastructure, including functions/devices and their subsystems, and relevant protocols operating at the link and network layers across multiple device. Therefore there will be a strong relationship between the abstracted view provided by this service model and the detailed configuration view that will be provided by specific configuration models for network elements.

The service component will take input from customer service model such as L3SM service model [RFC8299] or composed VPN service model and translate it into segment VPN in each domain and then further break down the segment VPN into detailed configuration view that will be provided by specific configuration models for network elements.

10. 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 [RFC5246].

The NETCONF access control model [RFC6536] 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:

o /composed-vpns/composed-vpn

  The entries in the list above include the whole composed vpn service configurations which the customer subscribes, and indirectly create or modify the PE,CE and ASBR device configurations. Unexpected changes to these entries could lead to service disruption and/or network misbehavior.

o /composed-vpns/composed-vpn/segment-vpn

  The entries in the list above include the access points configurations. As above, unexpected changes to these entries could lead to service disruption and/or network misbehavior.

o /composed-vpns/composed-vpn/access-point

The entries in the list above include the access points configurations. As above, unexpected changes to these entries could lead to service disruption and/or network misbehavior.

11. IANA Considerations

This document registers a URI in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested to be made:

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

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

This document registers two YANG modules in the YANG Module Names registry [RFC6020].

```
---------------------------------------------------------------------
           Name: ietf-composite-vpn-svc
           Namespace: urn:ietf:params:xml:ns:yang:ietf-composed-vpn-svc
           Prefix: composed-svc
           Reference: RFC xxxx
           Name: ietf-segmented-vpn
           Namespace: urn:ietf:params:xml:ns:yang:ietf-segment-vpn
           Prefix: segment-vpn
           Reference: RFC xxxx
---------------------------------------------------------------------
```

12. References

12.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", March 1997.

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

[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>.

[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>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <https://www.rfc-editor.org/info/rfc6536>.

[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>.

[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>.

[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>.

12.2. Informative References

[RFC1136] Hares, S. and D. Katz, "Administrative Domains and Routing Domains: A model for routing in the Internet", RFC 1136, DOI 10.17487/RFC1136, December 1989, <https://www.rfc-editor.org/info/rfc1136>.

[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>.

Appendix A. Acknowledges

Geng Liang,Congfeng Xie, Chen Rui, LiYa Zhang,Hui Deng contributed to an earlier version of [I-D.chen-opsawg-composite-vpn-dm]. We would like to thank the authors of that document on the operators' view for the PE-based VPN service configuration for material that assisted in thinking about this document.

Authors' Addresses

Roni Even
Huawei Technologies,Co.,Ltd
Tel Aviv
Israel

Email: roni.even@huawei.com

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

Email: lana.wubo@huawei.com

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

Email: bill.wu@huawei.com

YingCheng
China Unicom
No.21 Financial Street, XiCheng District
Beijing 100033
China

Email: chengying10@chinaunicom.cn

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 3, 2019

O. Friel
E. Lear
M. Pritikin
Cisco
M. Richardson
Sandelman Software Works

July 02, 2018

BRSKI over IEEE 802.11
draft-friel-brski-over-802dot11-01

Abstract

This document outlines the challenges associated with implementing Bootstrapping Remote Secure Key Infrastructures over IEEE 802.11 and IEEE 802.1x networks. Multiple options are presented for discovering and authenticating to the correct IEEE 802.11 SSID. This initial draft is a discussion document and no final recommendations are made on the recommended approaches to take.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Bootstrapping Remote Secure Key Infrastructures (BRSKI) [I-D.ietf-anima-bootstrapping-keyinfra] describes how a device can bootstrap against a local network using an Initial Device Identity X.509 [IEEE802.1AR] IDevID certificate that is pre-installed by the vendor on the device in order to obtain an [IEEE802.1AR] LDevID. The BRSKI flow assumes the device can obtain an IP address, and thus assumes the device has already connected to the local network. Further, the draft states that BRSKI use of IDevIDs:

> allows for alignment with [IEEE802.1X] network access control methods, its use here is for Pledge authentication rather than network access control. Integrating this protocol with network access control, perhaps as an Extensible Authentication Protocol (EAP) method (see [RFC3748], is out-of-scope.

The draft does not describe any mechanisms for how an [IEEE802.11] enabled device would discover and select a suitable [IEEE802.11] SSID when multiple SSIDs are available. A typical deployment scenario could involve a device begin deployed in a location were twenty or more SSIDs are being broadcast, for example, in a multi-tenanted building or campus where multiple independent organizations operate [IEEE802.11] networks.

In order to reduce the administrative overhead of installing new devices, it is desirable that the device will automatically discover and connect to the correct SSID without the installer having to manually provision any network information or credentials on the device. It is also desirable that the device does not discover, connect to, and automatically enroll with the wrong network as this could result in a device that is owned by one organization connecting to the network of a different organization in a multi-tenanted building or campus.

Additionally, as noted above, the BRSKI draft does not describe how BRSKI could potentially align with [IEEE802.1X] authentication mechanisms.

This document outlines multiple different potential mechanisms that would enable a bootstrapping device to choose between different available [IEEE802.11] SSIDs in order to execute the BRSKI flow. This document also outlines several options for how [IEEE802.11] networks enforcing [IEEE802.1X] authentication could enable the BRSKI flow, and describes the required device behaviour.

This document presents both [IEEE802.11] mechanisms and Wi-Fi Alliance (WFA) mechanisms. An important consideration when determining what the most appropriate solution to device onboarding should be is what bodies need to be involved in standardisation efforts: IETF, IEEE and/or WFA.

1.1. Terminology

IEEE 802.11u: an amendment to the IEEE 802.11-2007 standard to add features that improve interworking with external networks.

ANI: Autonomic Networking Infrastructure

ANQP: Access Network Query Protocol

AP: IEEE 802.11 Access Point

CA: Certificate Authority

EAP: Extensible Authentication Protocol

EST: Enrollment over Secure Transport

HotSpot 2.0 / HS2.0: An element of the Wi-Fi Alliance Passpoint certificatoin program that enables cell phones to automatically discover capabilities and enroll into IEEE 802.11 guest networks (hotspots).

IE: Information Element

IDevID: Initial Device Identifier

LDevID: Locally Significant Device Identifier

OI: Organization Identifier

MASA: BRSKI Manufacturer Authorized Signing Authority service

SSID: IEEE 802.11 Service Set Identifier

STA: IEEE 802.11 station

WFA: Wi-Fi Alliance

WLC: Wireless LAN Controller

WPA/WPA2: Wi-Fi Protected Access / Wi-Fi Protected Access version 2

WPS: Wi-Fi Protected Setup

2. Discovery and Authentication Design Considerations

2.1. Incorrect SSID Discovery

As will be seen in the following sections, there are several discovery scenarios where the device can choose an incorrect SSID and attempt to join the wrong network. For example, the device is being deployed by one organization in a multi-tenant building, and chooses to connect to the SSID of a neighbor organization. The device is dependent upon the incorrect network rejecting its BRSKI enrollment attempt. It is possible that the device could end up enrolled with the wrong network.

2.1.1. Leveraging BRSKI MASA

2.1.1.1. Prevention

BRSKI allows optional sales channel integration which could be used to ensure only the "correct" network can claim the device. In theory, this could be achieved if the BRSKI MASA service has explicit knowledge of the network where every single device will be deployed. After connecting to the incorrect SSID and possibly authenticating to the network, the device would present network TLS information in its voucher-request, and the MASA server would have to reject the request based on this network TLS information and not issue a voucher. The device could then reject that SSID and attempt to bootstrap against the next available SSID.

This could possibly be acheieved via sales channel integration, where devices are tracked through the supply chain all the way from manufacturer factory to target deployment network operator. In practice, this approach may be challenging to deploy as it may be extremely difficult to implement this tightly coupled sales channel integration and ensure that the MASA actually has accurate deployment network information.

An alternative to sales channel integration is to provide the device owners with a, possibly authenticated, interface or API to the MASA service whereby they would have to explicitly claim devices prior to the MASA issuing vouchers for that device. There are similar problems with this approach, as there could be a complex sales and channel partner chain between the MASA service operator and the device operator who owns and deploys the device. This could make exposure of APIs by the MASA operator to the device operator untenable.

2.1.1.2. Detection

If a device connects to the wrong network, the correct network operator could detect this after the fact by integration with MASA and checking audit logs for the device. The MASA audit logs should indicate all networks that have been issued vouchers for a specific device. This mechanism also relies on the correct network operater having a list, bill or materials, or similar of all device identities that should be connecting to their network in order to check MASA logs for devices that have not come online, but are known to be physically deployed.

2.1.2. Relying on the Network Administrator

An obvious mechanism is to rely on network administrators to be good citizens and explicitly reject devices that attempt to bootstrap against the wrong network. This is not guaranteed to work for two main reasons:

- o Some network administrators will configure an open policy on their network. Any device that attempts to connect to the network will be automatically granted access.

- o Some network administrators will be bad actors and will intentionally attempt to onboard devices that they do not own but that are in range of their networks.

2.1.3. Requiring the Network to Demonstrate Knowledge of Device

Protocols such as the WFA Device Provisioning Profile [DPP] require that a network provisoining entity demonstrate knowledge of device information such as the device’s bootstrapping public key prior to the device attempting to connect to the network. This gives a higher level of confidence to the device that it is connecting to the correct SSID. These mechanisms could leverage a key that is printed on the device label, or included in a sales channel bill of materials. The security of these types of key distribution mechanisms relies on keeping the device label or bill of materials content from being compromised prior to device installation.

2.2. IEEE 802.11 Authentication Mechanisms

[IEEE802.11i] allows an SSID to advertise different authentication mechanisms via the AKM Suite list in the RSNE. A very brief introduction to [IEEE802.11i] is given in the appendices. An SSID could advertise PSK or [IEEE802.1X] authentication mechanisms. When a network operator needs to enforce two different authentication mechanisms, one for pre-BRSKI devices and one for post-BRSKI devices, the operator has two options:

- o configure two SSIDs with the same SSID string value, each one advertising a different authentication mechanism

- o configure two different SSIDs, each with its own SSID string value, with each one advertising a different authentication mechanism

If devices have to be flexible enough to handle both options, then this adds complexity to the device firmware and internal state machines. Similarly, if network infrastructure (APs, WLCs, AAAs) potentially needs to support both options, then this adds complexity to network infrastructure configuration flexibility, software and state machines. Consideration must be given to the practicalities of implementation for both devices and network infrastructure when designing the final bootstrap mechanism and aligning [IEEE802.11], [IEEE802.1X] and BRSKI protocol interactions.

Devices should be flexible enough to handle potential options defined by any final draft. When discovering a pre-BRSKI SSID, the device should also discover the authentication mechanism enforced by the SSID that is advertising BRSKI support. If the device supports the authentication mechanism being advertised, then the device can connect to the SSID in order to initiate the BRSKI flow. For example, the device may support [IEEE802.1X] as a pre-BRSKI authentication mechanism, but may not support PSK as a pre-BRSKI authentication mechanism.

Once the device has completed the BRKSI flow and has obtained an LDevID, a mechanism is needed to tell the device which SSID to use for post-BRSKI network access. This may be a different SSID to the pre-BRSKI SSID. The mechanism by which the post-BRSKI SSID is advertised to the device is out-of-scope of this version of this document.

2.2.1. IP Address Assignment Considerations

If a device has to perform two different authentications, one for pre-BRSKI and one for post-BRSKI, network policy will typically assign the device to different VLANs for these different stages, and may assign the device different IP addresses depending on which network segment the device is assigned to. This could be true even if a single SSID is used for both pre-BRSKI and post-BRSKI connections. Therefore, the bootstrapping device may need to completely reset its network connection and network software stack, and obtain a new IP address between pre-BRSKI and post-BRSKI connections.

2.3. Client and Server Implementations

When evaluating all possible SSID discovery mechanism and authentication mechanisms outlined in this document, consideration must be given to the complexity of the required client and server implementation and state machines. Consideration must also be given to the network operator configuration complexity if multiple permutations and combinations of SSID discovery and network authentication mechanisms are possible.

3. Potential SSID Discovery Mechanisms

This section outlines multiple different mechanisms that could potentially be leveraged that would enable a bootstrapping device to choose between multiple different available [IEEE802.11] SSIDs. As noted previously, this draft does not make any final recommendations.

The discovery options outlined in this document include:

- Well-known BRSKI SSID
- [IEEE802.11aq]
- [IEEE802.11] Vendor Specific Information Element
- Reusing Existing [IEEE802.11u] Elements
- [IEEE802.11u] Interworking Information - Internet
- Define New [IEEE802.11u] Extensions
- Wi-Fi Protected Setup
- Define and Advertise a BRSKI-specific AKM in RSNE
- Wi-Fi Device Provisioning Profile

These mechanisms are described in more detail in the following sections.

3.1. Well-known BRSKI SSID

A standardized naming convention for SSIDs offering BRSKI services is defined such as:

- BRSKI%ssidname

Where:

- BRSKI: is a well-known prefix string of characters. This prefix string would be baked into device firmware.

- %: is a well known delimiter character. This delimiter character would be baked into device firmware.

- ssidname: is the freeform SSID name that the network operator defines.

Device manufacturers would bake the well-known prefix string and character delimiter into device firmware. Network operators configuring SSIDs which offer BRSKI services would have to ensure that the SSID of those networks begins with this prefix. On bootstrap, the device would scan all available SSIDs and look for ones with this given prefix.

If multiple SSIDs are available with this prefix, then the device could simply round robin through these SSIDs and attempt to start the BRSKI flow on each one in turn until it succeeds.

This mechanism suffers from the limitations outlined in Section 2.1 - it does nothing to prevent a device enrolling against an incorrect network.

Another issue with defining a specific naming convention for the SSID is that this may require network operators to have to deploy a new SSID. In general, network operators attempt to keep the number of unique SSIDs deployed to a minimum as each deployed SSID eats up a percentage of available air time and network capacity. A good discussion of SSID overhead and an SSID overhead [calculator] is available.

3.2. IEEE 802.11aq

[IEEE802.11aq] is currently being worked by the IEEE, but is not yet finalized, and is not yet supported by any vendors in shipping product. [IEEE802.11aq] defines new elements that can be included in [IEEE802.11] Beacon, Probe Request and Probe Response frames, and defines new elements for ANQP frames.

The extensions allow an AP to broadcast support for backend services, where allowed services are those registered in the [IANA] Service Name and Transport Protocol Port Number Registry. The services can be advertised in [IEEE802.11] elements that include either:

- o SHA256 hashes of the registered service names
- o a bloom filter of the SHA256 hashes of the registered service names

Bloom filters simply serve to reduce the size of Beacon and Probe Response frames when a large number of services are advertised. If a bloom filter is used by the AP, and a device discovers a potential service match in the bloom filter, then the device can query the AP for the full list of service name hashes using newly defined ANQP elements.

If BRSKI were to leverage [IEEE802.11aq], then the [IEEE802.11aq] specification would need to be pushed and supported, and a BRSKI service would need to be defined in [IANA].

This mechanism suffers from the limitations outlined in Section 2.1 - it does nothing to prevent a device enrolling against an incorrect network.

3.3. IEEE 802.11 Vendor Specific Information Element

[IEEE802.11] defines Information Element (IE) number 221 for carrying Vendor Specific information. The purpose of this document is to define an SSID discovery mechanism that can be used across all devices and vendors, so use of this IE is not an appropriate long term solution.

3.4. Reusing Existing IEEE 802.11u Elements

[IEEE802.11u] defines mechanisms for interworking. An introduction to [IEEE802.11u] is given in the appendices. Existing IEs in [IEEE802.11u] include:

- o Roaming Consortium IE
- o NAI Realm IE

These existing IEs could be used to advertise a well-known, logical service that devices implicitly know to look for.

In the case of NAI Realm, a well-known service name such as "_bootstrapks" could be defined and advertised in the NAI Realm IE. In the case of Roaming Consortium, a well-known Organization Identifier (OI) could be defined and advertised in the Roaming Consortium IE.

Device manufacturers would bake the well-known NAI Realm or Roaming Consortium OI into device firmware. Network operators configuring SSIDs which offer BRSKI services would have to ensure that the SSID offered this NAI Realm or OI. On bootstrap, the device would scan all available SSIDs and use ANQP to query for NAI Realms or Roaming Consortium OI looking for a match.

The key concept with this proposal is that BRSKI uses a well-known NAI Realm name or Roaming Consortium OI more as a logical service advertisement rather than as a backhaul internet provider advertisement. This is conceptually very similar to what [IEEE802.11aq] is attempting to achieve.

Leveraging NAI Realm or Roaming Consortium would not require any [IEEE802.11] specification changes, and could possibly be defined by this IETF draft. Note that the authors are not aware of any currently defined IETF or IANA namespaces that define NAI Realms or OIs.

Additionally (or alternatively...) as NAI Realm includes advertising the EAP mechanism required, if a new EAP-BRSKI were to be defined, then this could be advertised. Devices could then scan for an NAI Realm that enforced EAP-BRSKI, and ignore the realm name.

This mechanism suffers from the limitations outlined in Section 2.1 - it does nothing to prevent a device enrolling against an incorrect network.

Additionally, as the IEEE is attempting to standardize logical service advertisement via [IEEE802.11aq], [IEEE802.11aq] would seem to be the more appropriate option than overloading an existing IE. However, it is worth noting that configuration of these IEs is supported today by WLCs, and this mechanism may be suitable for demonstrations or proof-of-concepts.

3.5. IEEE 802.11u Interworking Information - Internet

It is possible that an SSID may be configured to provide unrestricted and unauthenticated internet access. This could be advertised in the Interworking Information IE by including:

- o internet bit = 1

- o ASRA bit = 0

If such a network were discovered, a device could attempt to use the BRSKI well-known vendor cloud Registrar. Possibly this could be a default fall back mechanism that a device could use when determining which SSID to use.

3.6. Define New IEEE 802.11u Extensions

Of the various elements currently defined by [IEEE802.11u] for potentially advertising BRSKI, NAI Realm and Roaming Consortium IE are the two existing options that are a closest fit, as outlined above. Another possibility that has been suggested in the IETF mailers is defining an extension to [IEEE802.11u] specifically for advertising BRSKI service capability. Any extensions should be included in Beacon and Probe Response frames so that devices can discover BRSKI capability without the additional overhead of having to explicitly query using ANQP.

[IEEE802.11aq] appears to be the proposed mechanism for generically advertising any service capability, provided that service is registered with [IANA]. It is probably a better approach to encourage adoption of [IEEE802.11aq] and register a service name for BRSKI with [IANA] rather than attempt to define a completely new BRSKI-specific [IEEE802.11u] extension.

3.7. Wi-Fi Protected Setup

Wi-Fi Protected Setup (WPS) only works with Wi-Fi Protected Access (WPA) and WPA2 when in Personal Mode. WPS does not work when the network is in Enterprise Mode enforcing [IEEE802.1X] authentication. WPS is intended for consumer networks and does not address the security requirements of enterprise or IoT deployments.

3.8. Define and Advertise a BRSKI-specific AKM in RSNE

[IEEE802.11i] introduced the RSNE element which allows an SSID to advertise multiple authentication mechanisms. A new Authentication and Key Management (AKM) Suite could be defined that indicates the STA can use BRSKI mechanisms to authenticate against the SSID. The authentication handshake could be an [IEEE802.1X] handshake, possibly leveraging an EAP-BRSKI mechanism, the key thing here is that a new AKM is defined and advertised to indicate the specific BRSKI-capable EAP method that is supported by [IEEE802.1X], as opposed to the current [IEEE802.1X] AKMs which give no indication of the supported EAP mechanisms. It is clear that such method would limit the SSID to BRSKI-supporting clients. This would require an additional SSID specifically for BRSKI clients.

3.9. Wi-Fi Device Provisioning Profile

The [DPP] specification defines how an entity that is already trusted by a network can assist an untrusted entity in enrolling with the network. The description below assumes the [IEEE802.11] network is in infrastructure mode. DPP introduces multiple key roles including:

- o Configurator: A logical entity that is already trusted by the network that has capabilities to enroll and provision devices called Enrollees. A Configurator may be a STA or an AP.
- o Enrollee: A logical entity that is being provisioned by a Configurator. An Enrollee may be a STA or an AP.
- o Initiator: A logical entity that initiates the DPP Authentication Protocol. The Initiator may be the Configurator or the Enrollee.
- o Responder: A logical entity that responds to the Initiator of the DPP Authentication Protocol. The Responder may be the Configurator or the Enrollee.

In order to support a plug and play model for installation of devices, where the device is simply powered up for the first time and automatically discovers the network without the need for a helper or supervising application, for example an application running on a smart cell phone or tablet that performs the role of Configurator, then this implies that the AP must perform the role of the Configurator and the device or STA performs the role of Enrollee. Note that the AP may simply proxy DPP messages through to a backend WLC, but from the perspective of the device, the AP is the Configurator.

The DPP specification also mandates that the Initiator must be bootstrapped the bootstrapping public key of the Responder. For BRSKI purposes, the DPP bootstrapping public key will be the [IEEE802.1AR] IDevID of the device. As the boostrapping device cannot know in advance the bootstrapping public key of a specific operators network, this implies that the Configurator must take on the role of the Initiator. Therefore, the AP must take on the roles of both the Configurator and the Initiator.

More details to be added...

4. Potential Authentication Options

When the bootstrapping device determines which SSID to connect to, there are multiple potential options available for how the device authenticates with the network while bootstrapping. Several options are outlined in this section. This list is not exhaustive.

At a high level, authentication can generally be split into two phases using two different credentials:

- o Pre-BRSKI: The device can use its [IEEE802.1AR] IDevID to connect to the network while executing the BRSKI flow

- o Post-BRSKI: The device can use its [IEEE802.1AR] LDevID to connect to the network after completing BRSKI enrollment

The authentication options outlined in this document include:

- o Unauthenticated Pre-BRSKI and EAP-TLS Post-BRSKI

- o PSK or SAE Pre-BRSKI and EAP-TLS Post-BRSKI

- o MAC Address Bypass Pre-BRSKI and EAP-TLS Post-BRSKI

- o EAP-TLS Pre-BRSKI and EAP-TLS Post-BRSKI

- o New TEAP BRSKI mechanism

- o New [IEEE802.11] Authentication Algorithm for BRSKI and EAP-TLS Post-BRSKI

- o New [IEEE802.1X] EAPOL-Announcements to encapsulate BRSKI prior to EAP-TLS Post-BRSKI

These mechanisms are described in more detail in the following sections. Note that any mechanisms leveraging [IEEE802.1X] are [IEEE802.11] MAC layer authentication mechanisms and therefore the SSID must advertise WPA2 capability.

When evaluating the multiple authentication options outlined below, care and consideration must be given to the complexity of the software state machine required in both devices and services for implementation.

4.1. Unauthenticated Pre-BRSKI and EAP-TLS Post-BRSKI

The device connects to an unauthenticated network pre-BRSKI. The device connects to a network enforcing EAP-TLS post-BRSKI. The device uses its LDevID as the post-BRSKI EAP-TLS credential.

To be completed..

4.2. PSK or SAE Pre-BRSKI and EAP-TLS Post-BRSKI

The device connects to a network enforcing PSK pre-BRSKI. The mechanism by which the PSK is provisioned on the device for pre-BRSKI authentication is out-of-scope of this version of this document. The device connects to a network enforcing EAP-TLS post-BRSKI. The device uses the LDevID obtained via BRSKI as the post-BRSKI EAP-TLS credential.

When the device connects to the post-BRSKI network that is enforcing EAP-TLS, the device uses its LDevID as its credential. The device should verify the certificate presented by the server during that EAP-TLS exchange against the trusted CA list it obtained during BRSKI.

If the [IEEE802.1X] network enforces a tunneled EAP method, for example [RFC7170], where the device must present an additional credential such as a password, the mechanism by which that additional credential is provisioned on the device for post-BRSKI authentication is out-of-scope of this version of this document. NAI Realm may be used to advertise the EAP methods being enforced by an SSID. It is to be determined if guidelines should be provided on use of NAI Realm for advertising EAP method in order to streamline BRSKI.

4.3. MAC Address Bypass Pre-BRSKI and EAP-TLS Post-BRSKI

Many AAA server state machine logic allows for the network to fallback to MAC Address Bypass (MAB) when initial authentication against the network fails. If the device does not present a valid credential to the network, then the network will check if the device’s MAC address is whitelisted. If it is, then the network may grant the device access to a network segment that will allow it to complete the BRSKI flow and get provisioned with an LDevID. Once the device has an LDevID, it can then reauthenticate against the network using its EAP-TLS and its LDevID.

4.4. EAP-TLS Pre-BRSKI and EAP-TLS Post-BRSKI

The device connects to a network enforcing EAP-TLS pre-BRSKI. The device uses its IDevID as the pre-BRSKI EAP-TLS credential. The device connects to a network enforcing EAP-TLS post-BRSKI. The device uses its LDevID as the post-BRSKI EAP-TLS credential.

When the device connects to a pre-BRSKI network that is enforcing EAP-TLS, the device uses its IDevID as its credential. The deivce should not attempt to verify the certificate presented by the server during that EAP-TLS exchange, as it has not yet discovered the local domain trusted CA list.

When the device connects to the post-BRSKI network that is enforcing EAP-TLS, the device uses its LDevID as its credential. The device should verify the certificate presented by the server during that EAP-TLS exchange against the trusted CA list it obtained during BRSKI.

Again, if the post-BRSKI network enforces a tunneled EAP method, the mechanism by which that second credential is provisioned on the device is out-of-scope of this version of this document.

4.5. New TEAP BRSKI mechanism

New TEAP TLVs are defined to transport BRSKI messages inside an outer EAP TLS tunnel such as TEAP [RFC7170]. [I-D.lear-eap-teap-brski] outlines a proposal for how BRSKI messages could be transported inside TEAP TLVs. At a high level, this enables the device to obtain an LDevID during the Layer 2 authentication stage. This has multiple advantages including:

- o avoids the need for the device to potentially connect to two different SSIDs during bootstrap
- o the device only needs to handle one authentication mechanism during bootstrap
- o the device only needs to obtain one IP address, which it obtains after BRSKI is complete
- o avoids the need for the device to have to disconnect from the network, reset its network stack, and reconnect to the network
- o potentially simplifies network policy configuration

There are two suboptions to choose from when tunneling BRSKI messages inside TEAP:

- o define new TLVs for transporting BRSKI messages inside the TEAP tunnel
- o define a new EAP BRSKI method type that is tunneled within the outer TEAP method

This section assumes that new TLVs are defined for transporting BRSKI messages inside the TEAP tunnel and that a new EAP BRSKI method type is not defined.

The device discovers and connects to a network enforcing TEAP. A high level TEAP with BRSKI extensions flow would look something like:

- Device starts the EAP flow by sending the EAP TLS ClientHello message

- EAP server replies and includes CertificateRequest message, and may specify certificate_authorities in the message

- if the device has an LDevID and the LDevID issuing CA is allowed by the certificate_authorities list (i.e. the issuing CA is explicitly included in the list, or else the list is empty) then the device uses its LDevID to establish the TLS tunnel

- if the device does not have an LDevID, or certificate_authorities prevents it using its LDevID, then the device uses its IDevID to establish the TLS tunnel

- if certificate_authorities prevents the device from using its IDevID (and its LDevID if it has one) then the device fails to connect

The EAP server continues with TLS tunnel establishment:

- if the device certificate is invalid or expired, then the EAP server fails the connection request.

- if the device certificate is valid but is not allowed due to a configured policy on the EAP server, then the EAP server fails the connection request

- if the device certificate is accepted, then the EAP server establishes the TLS tunnel and starts the tunneled EAP-BRSKI procedures

At this stage, the EAP server has some policy decisions to make:

- if network policy indicates that the device certificate is sufficient to grant network access, whether it is an LDevID or an IDevID, then the EAP server simply initiates the Crypto-Binding TLV and 'Success' Result TLV exchange. The device can now obtain an IP address and connect to the network.

- the EAP server may instruct the device to initialise a full BRSKI flow. Typically, the EAP server will instruct the device to initialize a BRSKI flow when it presents an IDevID, however, the EAP server may instruct the device to initialize a BRSKI flow even if it presented a valid LDevID. The device sends all BRSKI messages, for example 'requestvoucher', inside the TLS tunnel using new TEAP TLVs. Assuming the BRSKI flow completes successfully and the device is issued an LDevID, the EAP server completes the exchange by initiating the Crypto-Binding TLV and 'Success' Result TLV exchange.

Once the EAP flow has successfully completed, then:

- o network policy will automatically assign the device to the correct network segment
- o the device obtains an IP address
- o the device can access production service

It is assumed that the device will automatically handle LDevID certificate reenrolment via standard EST [RFC7030] outside the context of the EAP tunnel.

An item to be considered here is what information is included in Beacon or Probe Response frames to explicitly indicate that [IEEE802.1X] authentication using TEAP supporting BRSKI extensions is allowed. Currently, the RSNE included in Beacon and Probe Response frames can only indicate [IEEE802.1X] support.

4.6. New IEEE 802.11 Authentication Algorithm for BRSKI and EAP-TLS Post-BRSKI

[IEEE802.11] supports multiple authentication algorithms in its Authentication frame including:

- o Open System
- o Shared Key
- o Fast BSS Transition
- o Simultaneous Authentication of Equals

Shared Key authentication is used to indicate that the legacy WEP authentication mechanism is to be used. Simultaneous Authentication of Equals is used to indicate that the Dragonfly-based shared passphrase authentication mechanism introduced in [IEEE802.11s] is to be used. One thing that these two methods have in common is that a series of handshake data exchanges occur between the device and the AP as elements inside Authentication frames, and these Authentication exchanges happen prior to [IEEE802.11] Association.

It would be possible to define a new Authentication Algorithm and define new elements to encapsulate BRSKI messages inside Authentication frames. For example, new elements could be defined to encapsulate BRSKI requestvoucher, voucher and voucher telemetry JSON messages. The full BRSKI flow completes and the device gets issued an LDevID prior to associating with an SSID, and prior to doing full [IEEE802.1X] authentication using its LDevID.

The high level flow would be something like:

- o SSID Beacon / Probe Response indicates in RSNE that it supports BRSKI based Authentication Algorithm
- o SSIDs could also advertise that they support both BRSKI based Authentication and [IEEE802.1X]
- o device discovers SSID via suitable mechanism
- o device completes BRSKI by sending new elements inside Authentication frames and obtains an LDevID
- o device associates with the AP
- o device completes [IEEE802.1X] authentication using its LDevID as credential for EAP-TLS or TEAP

4.7. New IEEE 802.1X EAPOL-Announcements to encapsulate BRSKI and EAP-TLS Post-BRSKI

[IEEE802.1X] defines multiple EAPOL packet types, including EAPOL-Announcement and EAPOL-Announcement-Req messages. EAPOL-Annoncement and EAPOL-Announcement-Req messages can include multiple TLVs. EAPOL-Annoncement messages can be sent prior to starting any EAP authentication flow. New TLVs could be defined to encapsulate BRSKI messages inside EAPOL-Announcement and EAPOL-Announcement-Req TLVs. For example, new TLVs could be defined to encapsulate BRSKI requestvoucher, voucher and voucher telemetry JSON messages. The full BRSKI flow could complete inside EAPOL-Announcement exchanges prior to sending EAPOL-Start or EAPOL-EAP messages.

The high level flow would be something like:

- o SSID Beacon / Probe Response indicates somehow in RSNE that it supports [IEEE802.1X] including BRSKI extensions.
- o device connects to SSID and completes standard Open System Authentication and Association
- o device starts [IEEE802.1X] EAPOL flow and uses new EAPOL-Announcement frames to encapsulate and complete BRSKI flow to obtain an LDevID

- o device completes [IEEE802.1X] authentication using its LDevID as credential for EAP-TLS or TEAP

5. IANA Considerations

[[TODO]]

6. Security Considerations

[[TODO]]

7. Informative References

[calculator]
Revolution Wi-Fi, "SSID Overhead Calculator", n.d., <http://www.revolutionwifi.net/revolutionwifi/p/ssid-overhead-calculator.html>.

[DPP] Wi-Fi Alliance, "Wi-Fi Device Provisioning Protocol", n.d., <https://www.wi-fi.org/file/wi-fi-device-provisioning-protocol-dpp-draft-technical-specification-v0023>.

[I-D.ietf-anima-bootstrapping-keyinfra]
Pritikin, M., Richardson, M., Behringer, M., Bjarnason, S., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructures (BRSKI)", draft-ietf-anima-bootstrapping-keyinfra-16 (work in progress), June 2018.

[I-D.lear-eap-teap-brski]
Lear, E., Friel, O., and N. Cam-Winget, "Bootstrapping Key Infrastructure over EAP", draft-lear-eap-teap-brski-00 (work in progress), June 2018.

[IANA] Internet Assigned Numbers Authority, "Service Name and Transport Protocol Port Number Registry", n.d., <https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml>.

[IEEE802.11]
IEEE, ., "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", 2016.

[IEEE802.11aq]
IEEE, ., "802.11 Amendment 5 Pre-Association Discovery", 2017.

[IEEE802.11i]
IEEE, ., "802.11 Amendment 6 Medium Access Control (MAC) Security Enhancements", 2004.

[IEEE802.11s]
IEEE, ., "802.11 Amendment 10 Mesh Networking", 2011.

[IEEE802.11u]
IEEE, ., "802.11 Amendment 9 Interworking with External Networks", 2011.

[IEEE802.1AR]
IEEE, ., "Secure Device Identity", 2017.

[IEEE802.1X]
IEEE, ., "Port-Based Network Access Control", 2010.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, Ed., "Extensible Authentication Protocol (EAP)", RFC 3748, DOI 10.17487/RFC3748, June 2004, <https://www.rfc-editor.org/info/rfc3748>.

[RFC4282] Aboba, B., Beadles, M., Arkko, J., and P. Eronen, "The Network Access Identifier", RFC 4282, DOI 10.17487/RFC4282, December 2005, <https://www.rfc-editor.org/info/rfc4282>.

[RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed., "Enrollment over Secure Transport", RFC 7030, DOI 10.17487/RFC7030, October 2013, <https://www.rfc-editor.org/info/rfc7030>.

[RFC7170] Zhou, H., Cam-Winget, N., Salowey, J., and S. Hanna, "Tunnel Extensible Authentication Protocol (TEAP) Version 1", RFC 7170, DOI 10.17487/RFC7170, May 2014, <https://www.rfc-editor.org/info/rfc7170>.

Appendix A. IEEE 802.11 Primer

A.1. IEEE 802.11i

802.11i-2004 is an IEEE standard from 2004 that improves connection security. 802.11i-2004 is incorporated into 802.11-2014. 802.11i defines the Robust Security Network IE which includes information on:

- o Pairwise Cipher Suites (WEP-40, WEP-104, CCMP-128, etc.)

- o Authentication and Key Management Suites (PSK, 802.1X, etc.)

The RSN IEs are included in Beacon and Probe Response frames. STAs can use this frame to determine the authentication mechanisms offered by a particular AP e.g. PSK or 802.1X.

A.2. IEEE 802.11u

802.11u-2011 is an IEEE standard from 2011 that adds features that improve interworking with external networks. 802.11u-2011 is incorporated into 802.11-2016.

STAs and APs advertise support for 802.11u by setting the Interworking bit in the Extended Capabilities IE, and by including the Interworking IE in Beacon, Probe Request and Probe Response frames.

The Interworking IE includes information on:

- o Access Network Type (Private, Free public, Chargeable public, etc.)

- o Internet bit (yes/no)

- o ASRA (Additional Step required for Access - e.g. Acceptance of terms and conditions, On-line enrollment, etc.)

802.11u introduced Access Network Query Protocol (ANQP) which enables STAs to query APs for information not present in Beacons/Probe Responses.

ANQP defines these key IEs for enabling the STA to determine which network to connect to:

- o Roaming consortium IE: includes the Organization Identifier(s) of the roaming consortium(s). The OI is typically provisioned on cell phones by the SP, so the cell phone can automatically detect 802.11 networks that provide access to its SP’s consortium.

- o 3GPP Cellular Network IE: includes the Mobile Country Code (MCC) and Mobile Network Code (MNC) of the SP the AP provides access to.

- o Network Access Identifier Realm IE: includes [RFC4282] realm names that the AP provides access to (e.g. wifi.service-provider.com). The NAI Realm IE also includes info on the EAP type required to access that realm e.g. EAP-TLS.

- o Domain name IE: the domain name(s) of the local AP operator. Its purpose is to enable a STA to connect to a domain operator that may have a roaming agreement with STA’s Service Provider.

STAs can use one or more of the above IEs to make a suitable decision on which SSID to pick.

HotSpot 2.0 is an example of a specification built on top of 802.11u and defines 10 additional ANQP elements using the standard vendor extensions mechanisms defined in 802.11. It also defines a HS2.0 Indication element that is included in Beacons and Probe Responses so that STAs can immediately tell if an SSID supports HS2.0.

Authors' Addresses

Owen Friel
Cisco

Email: ofriel@cisco.com

Eliot Lear
Cisco

Email: lear@cisco.com

Max Pritikin
Cisco

Email: pritikin@cisco.com

Michael Richardson
Sandelman Software Works

Email: mcr+ietf@sandelman.ca

Network Working Group E. Lear
Internet-Draft O. Friel
Intended status: Standards Track Cisco Systems
Expires: April 23, 2019 October 20, 2018

Proof of Possession to Devices for Onboarding
draft-lear-brski-pop-00

Abstract

This memo specifies a RESTful interface for local deployments to demonstrate proof of possession to a device or to a manufacturer authorized signing authority (MASA). This covers the case where a MASA would not otherwise have knowledge of where a device is deployed, or when a MASA may not be required. Such knowledge is important to onboard certain classes of devices, such as those on IEEE 802.11 networks.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

[I-D.ietf-anima-bootstrapping-keyinfra] (BRSKI) specifies a means to provision credentials to be used as credentials to operationally access networks. In the initial model, the manufacturer authorized signing authority is assumed to either have knowledge of whether a device is intended to be provisioned on a particular network, or to be able to simply sign all requests. The necessary knowledge to handle the first case is not always easy to come by, and particularly useful to have when a device is trying to determine which network to join, when there is a choice. Such is the case with IEEE 802.11 networks, for example.

Absent that knowledge, should a MASA automatically issue a voucher, the device may onboard to the first BRSKI-aware network, which may well be the wrong one.

In addition, some manufacturers may prefer not to require the existence of a MASA. In these circumstances proof of possession to the device is required.

This memo specifies a RESTful request that devices and registrars employ as an alternative to [I-D.ietf-anima-bootstrapping-keyinfra], in which two additional optional objects may be specified. Three new objects are defined:

1. A simple binary claim that registrar administrator knows this device to belong on the particular deployment network. This object should be conveyed from the registrar to the MASA.

2. A cryptographic claim as such. This would typically be some sort of scanned label or information received as part of a bill of materials that contains some signed evidence of delivery of the

end device to the deployment. This option may be conveyed from the register to the MASA, or when the MASA needn't be contacted, to the device.

3. A statement indicating that the MASA server needn't be contacted at all, and that the device will accept a certificate with the cryptographic claim specified in this memo. This permits offline registration.

Note that this interface is optional. There may well be cases where a MASA already has sufficient knowledge to onboard a device to the correct network. Particularly where the manufacturer requires online registration, when such integration exists, the mechanisms defined in this memo SHOULD NOT be used, as they would be superfluous.

When this model is used, in order to avoid any interoperability problems, a new RESTful endpoint is defined as follows:

"/.well-known/est/request-voucher-with-possession"

The new endpoint is handled precisely as described in Section 5.2 of [I-D.ietf-anima-bootstrapping-keyinfra], with the exception voucher is formed as described below in Section 2.

If the device has indicated that the MASA server needn't be contacted, then the registrar may generate an unsigned voucher response. However, in this case, the registrar must include a valid claim object that has been hashed with an 8-32 bit nonce, immediately succeeded by a non-NULL-terminated key that is provided in UTF8 format. The response MUST be a voucher-brski-pop-request-artifact rather than a voucher-artifact.

2. The Yang Model

```
<CODE BEGINS>file "ietf-brski-possession@2018-10-11.yang"
module ietf-brski-possession {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-brski-possession";
  prefix mr;

  import ietf-restconf {
    prefix rc;
    description
      "This import statement is only present to access
       the yang-data extension defined in RFC 8040.";
    reference "RFC 8040: RESTCONF Protocol";
  }
  import ietf-voucher {
```

```
    prefix v;
    description "This module defines the format for a voucher,
        which is produced by a pledge's manufacturer or
        delegate (MASA) to securely assign a pledge to
        an 'owner', so that the pledge may establish a secure
        conn ection to the owner's network infrastructure";

    reference "RFC 8366: Voucher Profile for Bootstrapping Protocols";
  }

  import ietf-voucher-request {
    prefix rv;
    description
      "Voucher request is what we will augment";
    reference "draft-ietf-anima-bootstrapping-keyinfra";
  }

  organization
    "TBD";
  contact
    "Author: Eliot Lear
             <mailto:lear@cisco.com>";
  description
    "This module to provide additional information about
     how a device may be claimed by a particular deployment.
     The owner is asserting that this information has not merely
     been gleaned  directly in-band from the device,
     but rather he or she can confirm ownership independently.

     Copyright (c) 2018 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).

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

  revision 2018-10-11 {
    description
      "Initial version";
    reference "RFC XXXX: Proof of possession for BRSKI";
  }
  rc:yang-data voucher-brski-pop-request-artifact {
    uses rv:voucher-request-grouping {
```

```
      augment "voucher" {
        description
          "trying to add one more thing into this voucher.";
        leaf out-of-band-claim {
          when 'not(../no-masa-required) and not(../possession-claim)';
          type binary;
          description
            "If this value is true, then the adminsitrator of the
             registrar is claiming that the device being claimed
             has been purchased or otherwise acquired for this
             deployment, and that the information has not merely
             been automatically gleaned directly from the device.";
        }
        leaf possession-claim {
          when 'not(../no-masa-required) and not(../out-of-band-claim)';
          type string;
          description
            "In the context of a voucher-request, this node contains
             a naked key that the MASA will validate.  If valid, the
             MASA will sign a voucher.  The form of this key is left
             to the manufacturer, and is opaque to the registrar";
        }
        leaf no-masa-required {
          when 'not(../possession-claim)and not(../out-of-band-claim)';
          type binary;
          description
            "If true, then the device will not bother to validate
             the provisional TLS connection, but instead assume it
             to be valid.  Only the pledge may set this value.
             The registrar MUST have included the possession-claim
             object.";
        }
      }
    }
  }
  rc:yang-data voucher-with-pop-artifact {
    uses v:voucher-artifact-grouping {
      refine "voucher/pinned-domain-cert" {
        mandatory false;
        }
      refine "voucher/assertion" {
        mandatory false;
        }
      augment "voucher" {
        description
          "Add leaf node for returning a hashed proof of
          possession.";
```

```
          leaf hashed-proof-of-possession {
            type binary;
            mandatory true;
            description
              "A hash of the provided nonce and a key obtained
               by the registrar.  The format is the nonce followed
               immediately by the key.";
          }

          leaf hash-type {
             type enumeration {
               enum SHA256 {
                 description
                   "The type of hash is SHA256.";
               }
             }
             description
               "If not present, assume SHA256.  Otherwise, whatever
                augmented value is present.  This is for algorithmic
                agility.";
           }
         }
       }
     }
}

<CODE ENDS>
```

3. Examples

TBD.

4. IANA Considerations

The following YANG name space should be registered:

o "urn:ietf:params:xml:ns:yang:ietf-brski-possession"

5. Security Considerations

There will be many.

6. Acknowledgments

None yet.

7. Changes from Earlier Versions

Draft -00:

- o Initial revision

8. Normative References

[I-D.ietf-anima-bootstrapping-keyinfra]
Pritikin, M., Richardson, M., Behringer, M., Bjarnason, S., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructures (BRSKI)", draft-ietf-anima-bootstrapping-keyinfra-16 (work in progress), June 2018.

Authors' Addresses

Eliot Lear
Cisco Systems
Richtistrasse 7
Wallisellen CH-8304
Switzerland

Phone: +41 44 878 9200
Email: lear@cisco.com

Owen Friel
Cisco Systems
170 W. Tasman Dr.
San Jose, CA 95134
United States

Email: ofriel@cisco.com

Network Working Group
Internet-Draft
Updates: RFC7170 (if approved)
Intended status: Standards Track
Expires: May 4, 2020

E. Lear
O. Friel
N. Cam-Winget
Cisco Systems
D. Harkins
HP Enterprise

November 01, 2019

TEAP Update and Extensions for Bootstrapping
draft-lear-eap-teap-brski-05

Abstract

In certain environments, in order for a device to establish any layer three communications, it is necessary for that device to be properly credentialed. This is a relatively easy problem to solve when a device is associated with a human being and has both input and display functions. It is less easy when the human, input, and display functions are not present. To address this case, this memo specifies extensions to the Tunnel Extensible Authentication Protocol (TEAP) method that leverages Bootstrapping Remote Secure Key Infrastructures (BRSKI) in order to provide a credential to a device at layer two. The basis of this work is that a manufacturer will introduce the device and the local deployment through cryptographic means. In this sense the same trust model as BRSKI is used.

Status of This Memo

Table of Contents

1. Introduction

[I-D.ietf-anima-bootstrapping-keyinfra] (BRSKI) specifies a means to provision credentials to be used as credentials to operationally access networks. It was designed as a standalone means where some limited access to an IP network is already available. This is not always the case. For example, IEEE 802.11 networks generally require authentication prior to any form of address assignment. While it is possible to assign an IP address to a device on some form of an open network, or to accept some sort of default credential to establish initial IP connectivity, the steps that would then follow might well require that the device is placed on a new network, requiring reseting all layer three parameters.

A more natural approach in such cases is to more tightly bind the provisioning of credentials with the authentication mechanism. One such way to do this is to make use of the Extensible Authentication Protocol (EAP) [RFC3748] and the Tunnel Extensible Authentication Protocol (TEAP) method [RFC7170]. Thus we define new TEAP Type-Length-Value (TLV) objects that can be used to transport the BRSKI protocol messages within the context of a TEAP TLS tunnel.

[RFC7170] discusses the notion of provisioning peers. Several different mechanisms are available. Section 3.8 of that document acknowledges the concept of not initially authenticating the outer TLS session so that provisioning may occur. In addition, exchange of multiple TLV messages between client and EAP server permits multiple provisioning steps.

1.1. Terminology

The reader is presumed to be familiar with EAP terminology as stated in [RFC3748]. In addition, the following terms are commonly used in this document.

- o BRSKI: Bootstrapping Remote Secure Key Infrastructures, as defined in [I-D.ietf-anima-bootstrapping-keyinfra]. The term is also used to refer to the flow described in that document.

- o EST: Enrollment over Secure Transport, as defined in [RFC7030].

- o Voucher: a signed JSON object as defined in [RFC8366].

2. TEAP BRSKI Architecture

The TEAP BRSKI architecture is illustrated in Section 3. The device talks to the TEAP server via the Authenticator using any compliant transport such as [IEEE8021X]. The architecture illustrated shows an Authenticator distinct from the TEAP server. This is a deployment optimization and when so deployed the communication between Authenticator and TEAP server is a AAA protocol such as RADIUS or DIAMETER.

The architecture illustrated shows a co-located TEAP server and BRSKI registrar. Not only are these two functions co-located, they MUST be the same entity. This ensures that the entity identified in the device's voucher request (the TEAP server) is the same entity that signs the voucher request (the registrar).

The registrar communicates with the BRSKI MASA service for the purposes of getting signed vouchers.

The registrar also communicates with a Certificate Authority in order to issue LDevIDs. The architecture shows the registrar and CA as being two logically separate entities, however the CA may be integrated into the registrar. The device is not explicitly aware of whether the CA and registrar functions are integrated.

```
+--------+     +---------+     +---------+     +------+
|        |     |         |     |  TEAP   |<--->| MASA |
|        |     | Authen- |     | server  |     +------+
| Device |<--->| ticator |<--->|         |
|        |     |         |     |  BRSKI  |     +------+
|        |     |         |     |Registrar|<--->|  CA  |
+--------+     +---------+     +---------+     +------+
```

3. BRSKI Bootstrap and Enroll Operation

This section summarises the current BRSKI operation. The BRSKI flow assumes the device has an IDevID and has a manufacturer installed trust anchor that can be used to validate the BRSKI voucher. The

BRSKI flow compromises serveral main steps from the perspective of the device:

- Step 1: Device discovers the registrar

- Step 2: Device establishes provisional TLS connection to registrar

- Step 3: Device sends voucher request message and receives signed voucher response

- Step 4: Device validates voucher and validates provisional TLS connection to registrar

- Step 5: Device downloads additional local domain CA information

- Step 6: Device downloads Certificate Signing Request (CSR) attributes

- Step 7: Device does a certificate enroll to obtain an LDevID

- Step 8: Device periodically reenrolls via EST to refresh its LDevID

Most of the operational steps require the device, and thus its internal state machine, to automatically complete the next step without being explicitly instructed to do so by the registrar. For example, the registrar does not explicitly tell the device to download additional local domain CA information, or to do an EST enroll to obtain an LDevID.

3.1. Discovery of Trusted MASA

BRSKI section 2.8 outlines how the Registrar discovers the correct MASA to connect with. BRSKI section 5.3 outlines how the Registrar can make policy decisions about which devices to trust.

Similar approaches are applicable for TEAP servers executing BRSKI. For example, the TEAP server may be configured with a list of trusted manufacturing CAs. During device bootstrap, only devices with an IDevID signed by a trusted manufacturing CA may be allowed to etablishes a TLS connection with the TEAP server, and the TEAP server could then extract the MASA URI from the device's IDevID.

3.2. Executing BRSKI in a TEAP Tunnel

This section outlines how the main BRSKI steps outlined above map to TEAP, and how BRSKI and enrollment can be accomplished inside a TEAP TLS tunnel. The following new TEAP TLVs are introduced:

o BRSKI-VoucherRequest

o BRSKI-Voucher

o CSR-Attributes

The following steps outline how the above BRSKI flow maps to TEAP.

o Step 1: Device discovers the registrar

When BRSKI is executed in a TEAP tunnel, the device exchanges BRSKI TLVs with the TEAP server. The discovery process for devices is therefore the standard wired or wireless LAN EAP server discovery process. The discovery processes outlined in section 4 of [I-D.ietf-anima-bootstrapping-keyinfra] are not required for initial discovery of the registrar.

o Step 2: Device establishes provisional TLS connection to registrar

The device establishes an outer TEAP tunnel with the TEAP server and does not validate the server certificate. The device presents its LDevID as its identity certificate if it has a valid LDevID, otherwise it presents its IDevID. The TEAP server validates the device's certificate using its implicit or explicit trust anchor database. If the device presents an IDevID it is verified against a database of trusted manufacturer certificates. Server policy may also be used to control which certificate the device is allowed present, as described in section {pki-certificate-authority-considerations}.

If the presented credential is sufficient to grant access, the TEAP server can return a TEAP Result TLV indicating success immediately. The device may still send a Request-Action TLV including a BRSKI-VoucherRequest TLV in response to the TEAP Result TLV if it does not have, but requires, provisioning of trust anchors for validating the TEAP server certificate. Note that no inner EAP method is required for this, only an exchange of TEAP TLVs.

[todo] Question: as the device wants the server to reply with a BRSKI-Voucher TLV, does it really send a Request-Action TLV containing a BRSKI-VoucherRequest TLV, or does it send a Request-Action TLV containing a BRSKI-Voucher TLV?? The TEAP draft is a bit ambiguous here. Normally, if one end sends a Request-Action including XXX-TLV, it means it wants the far end ot send an XXX-TLV...

[todo] Question: general TEAP protocol question: does the device have to send a Request-Action w/BRSKI-VoucherRequest or can it send a BRSKI-VoucherRequest on its own? I'm not clear on this.

If the TEAP server requires that the device execute a BRSKI flow, the server sends a Request-Action TLV that includes a BRSKI-VoucherRequest TLV. For example, if the device presented its IDevID but the TEAP server requires an LDevID.

[todo] Question: to nit pick, the server should send a Request-Action TLV including a PKCS#10 TLV to tell the client to enroll. How does the server really know that the client has the correct trust established (as previously received by a BRSKI-Voucher)? If the client sends an IDevID, does server always send a Request-Action including both BRSKI-VoucherRequest and PKCS#10 TLVs? Whats the client behaviour? I assume client can spontaneously send BRSKI-VoucherRequest and/or PCSK#10 without being explicitly instructed to. Just need to get the language correct here.

The TEAP server may also require the device to reenroll, for example, if the device presented a valid LDevID that is very closed to expiration. The server may instruct a device to reenroll by sending a Request-Action TLV that includes a zero byte length PKCS#10 TLV.

- Step 3: Device sends voucher request message and receives signed voucher response

The device sends a BRSKI-RequestVoucher TLV to the TEAP server. The TEAP server forwards the RequestVoucher message to the MASA server, and the MASA server replies with a signed voucher. The TEAP server sends a BRSKI-Voucher TLV to the device.

If the MASA server does not issue a signed voucher, the TEAP server sends an EAP-Error TLV with a suitable error code to the device.

For wireless devices in particular, it is important that the MASA server only return a voucher for devices known to be associated with a particular registrar. In this sense, success indicates that the device is on the correct network, while failure indicates the device should try to provision itself within wireless networks (e.g, go to the next SSID).

- Step 4: Device validates voucher and validates provisional TLS connection to registrar

The device validates the signed voucher using its manufacturer installed trust anchor, and uses the CA information in the voucher to validate the TLS connection to the TEAP server.

If the device fails to validate the voucher, then it sends a TEAP-Error TLV indicating failiure to the TEAP server.

Similarly, if the device validates the voucher, but fails to validate the provisional TLS connection, then it sends a TEAP-Error TLV indicating failure to the TEAP server. Note that the outer TLS tunnel has already been established, thus allowing the client to send a TEAP-Error TLV to the server inside that tunnel to indicate that it failed to verify the provisionally accepted outer TLS tunnel server identity.

- Step 5: Device downloads additional local domain CA information

On completion of the BRSKI flow, the device SHOULD send a Trusted-Server-Root TLV to the TEAP server in order to discover additional local domain CAs. This is equivalent to section [todo] from [I-D.ietf-anima-bootstrapping-keyinfra].

- Step 6: Device downloads CSR attributes

No later than the completion of step 5, server MUST send a CSR-Attributes TLV to peer server in order to discover the correct fields to include when it enrolls to get an LDevID.

- Step 7: Device does a certificate enroll to obtain an LDevID

When executing the BRSKI flow inside a TEAP tunnel, the device does not directly leverage EST when doing its initial enroll. Instead, the device uses the existing TEAP PKCS#10 and PCKS#7 TEAP mechanisms.

Once the BRSKI flow is complete, the device can now send a PKCS#10 TLV to enroll and request an LDevID. If the TEAP server instructed the device to start the BRSKI flow via a Request-Action TLV that includes a BRSKI-RequestVoucher TLV, then the device MUST send a PKCS#10 in order to start the enroll process. The TEAP server will handle the PKCS#10 and ultimately return a PKCS#7 including an LDevID to the device.

If the TEAP server granted the device access on completion of the outer TEAP TLS tunnel in step 2 without sending a Request-Action TLV, the device does not have to send a PKCS#10 to enroll.

At this point, the device is said to be provisioned for local network access, and may authenticate in the future via 802.1X with its newly acquired credentials.

- Step 8: Device periodically reenrolls to refresh its LDevID

When a device's LDevID is close to expiration, there are two options for re-enrollment in order to obtain a fresh LDevID. As outlined in Step 2 above, the TEAP server may instruct the device to reenroll by sending a Request-Action TLV including a PKCS#10 TLV. If the TEAP server explicilty instructs the device to reenroll via these TLV exchange, then the device MUST send a PKCS#10 to reenroll and request a fresh LDevID.

However, the device SHOULD reenroll if it determines that its LDevID is close to expiration without waiting for explicit instruction from the TEAP server. There are two options to do this.

Option 1: The device reenrolls for a new LDevID directly with the EST CA outside the context of the 802.1X TEAP flow. The device uses the registrar discovery mechanisms oulined in [I-D.ietf-anima-bootstrapping-keyinfra] to discover the registrar and the device sends the EST reenroll messages to the discovered registrar endpoint. No new TEAP TLVs are defined to facilitate discover of the registrar or EST endpoints inside the context of the TEAP tunnel.

Option 2: When the device is performing a periodic 802.1X authentication using its current LDevID, it reenrolls for a new LDevID by sending a PKCS#10 TLV inside the TEAP TLS tunnel.

4. PKI Certificate Considerations

There are multiple noteworthy PKI certificate handling considerations. These include:

- PKI CA handling when establishing the TEAP tunnel
- PKI CA handling establishing trust using BRSKI
- IDevID and LDevID expiration times
- Specifying LDevID Subject and Subject Alternative Names
- PKCS#10 retry handling

These are described in more detail here.

4.1. TEAP Tunnel Establishment

Because this method establishes a client identity, if the peer has not been previously bootstrapped, or otherwise cannot successfully authenticate, it will use a generic identity string of teap-bootstrap@TBD1 as its network access identifier (NAI).

BRSKI section 5.3 outlines the policy decisions a Registrar may make when deciding whether to accept connections from clients. Similarly, the TEAP server operator may configure a set of trusted CAs for validating incoming TLS connections from clients. The operator may want to 'allow any device from a specific vendor', or from a set of vendors, to access the network. Network operators may do this by restricting network access to clients that have a certificate signed by one of a small set of trusted manufacturer/supplier CAs.

When the client sends its ClientHello to initiate TLS tunnel establishment, it is possible for the TEAP server to restrict the certificates that the client can use for tunnel establishment by including a list of CA distinguished names in the certificate_authorities field in the CertificateRequest message. The client should only continue with the handshake if it has a certificate signed by one of the indicated CAs.

In practice, network operators will likely want to onboard devices from a large number of device manufacturers, with each manufacturer using a different root CA when issuing IDevIDs. If the number of different manufacturer root CAs is large, this could result in very large TLS handshake messages. Therefore, the TEAP server may send a CertificateRequest message and not specify any certificate_authorities, thus allowing the client present a certificate signed by any authority in its Certificate message.

If the client has both an IDevID and an LDevID, the client should present the LDevID in preference to its IDevID, if allowed by server policy.

Once the client has sent its TLS Finished message, the TEAP server can make a policy decision, based on the CA used to sign the client's certificate, on whether to establish the outer TLS tunnel or not.

The TEAP server may delegate policy decisions to the MASA or CA function. For example, the TEAP server may declare EAP success and grant network access if the client presents a valid LDevID signed by a trusted domain CA. However, if the client presents an IDevID signed by a trusted manufacturer CA, the TEAP server may establish the TLS tunnel but not declare EAP success and grant network access until the client successfully completes a BRSKI Voucher exchange and PKCS#10/PKCS#7 exchange inside that tunnel.

It is recommended that the client validate the certificate presented by the server in the server's Certificate message, but this may not be possible for clients that have not yet provisioned appropriate trust anchors. If the client is in the provisioning phase and has not yet completed a BRSKI flow, it will not have trust anchors installed yet, and thus will not be able to validate the server's certificate. The client must however note the certificate presented by the server for (i) inclusion in the BRSKI-RequestVoucher TLV and for (ii) validation once the client has discovered the local domain trust anchors.

If the client does not present a suitable certificate to the server, the server MUST terminate the connection and fail the EAP request. If the TEAP server is unable to validate the client's certificate using its implicit or explicit trust anchor database it MUST fail the EAP request.

On establishment of the outer TLS tunnel, the TEAP server will make a policy decision on next steps. Possible policy decisions include:

- o Option 1: Server grants client full network access and returns EAP-Success. This will typically happen when the client presents a valid LDevID. Network policy may grant client network access based on IDevID without requiring the device to enroll to obtain an LDevID.

- o Option 2: Server requires that client perform a full BRSKI flow, and then enroll to get an LDevID. This will typically happen when the client presents a valid IDevID and network policy requires all clients to have LDevIDs. The server sends a Request-Action TLV that includes a BRSKI-RequestVoucher TLV to the client to instruct it to start the BRSKI flow.

- o Option 3: Server requires that the client reenroll to obtain a new LDevID. This could happen when the client presents a valid LDevID that is very close to expiration time, or the server's policy requires an LDevID update. The server sends an Action-Request TLV including a PKCS#10 TLV to the client to instruct it to reenroll.

4.2. BRSKI Trust Establishment

If the server requires that client perform a full BRSKI flow, it sends a Request-Action TLV that includes a zero byte length BRSKI-RequestVoucher TLV to the client. The client sends a new BRSKI-RequestVoucher TLV to the server, which contains all data specified in [I-D.ietf-anima-bootstrapping-keyinfra] section 5.2. The client includes the server certificate it received in the server's Certificate message during outer TLS tunnel establishment in the proximity-registrar-cert field. The client signs the request using its IDevID.

The server includes all additional information as required by [I-D.ietf-anima-bootstrapping-keyinfra] section 5.4 and signs the request prior to forwarding to the MASA.

The MASA responds as per [I-D.ietf-anima-bootstrapping-keyinfra] section 5.5. The response may indicate failure and the server should react accordingly to failures by sending a failure response to the client, and failing the TEAP method.

If the MASA replies with a signed voucher and a successful result, the server then forwards this response to the client in a BRSKI-Voucher TLV.

When the client receives the signed voucher, it validates the signature using its built in trust anchor list, and extracts the pinned-domain-cert field. The client must use the CA included in the pinned-domain-cert to validate the certificate that was presented by the server when establishing the outer TLS tunnel. If this certificate validation fails, the client must fail the TEAP request and not connect to the network.

[TBD- based on client responses, the registrar sends a status update to the MASA]

4.3. Certificate Expiration Times

[IEEE8021AR] section 7.2.7.2 states:

> notAfter: The latest time a DevID is expected to be used. Devices possessing an IDevID are expected to operate indefinitely into the future and should use the value 99991231235959Z. Solutions verifying an IDevID are expected to accept this value indefinitely.

TEAP servers SHOULD follow the 802.1AR standard when validating IDevIDs.

TEAP servers SHOULD reject LDevIDs with expired certificates and SHOULD NOT allow clients to connect with recently expired LDevIDs. If a client presents a recently expired LDevID it SHOULD be forced to authenticate using its IDevID and then reenroll to obtain a valid LDevID.

4.4. LDevID Subject and Subject Alternative Names

BRSKI section 5.9.2 specifies that the pledge MUST send a CSR Attributes request to the registrar. The registrar MAY use this mechanism to instruct the pledge about the identities it should include in the CSR request it sends as part of enrollment. The registrar may use this mechanism to tell the pledge what Subject or Subject Alternative Name identity information to include in its CSR request. This can be useful if the Subject must have a specific value in order to complete enrollment with the CA.

4.5. PKCS#10 Retry Handling

They will be scenarios where the TEAP server is willing to handle a PKCS#10 request from a client and issue a certificate via a PKCS#7 response, however, the TEAP server is unable to immediately completely the request and needs to instruct the client to retry later after a specified time interval.

A new Retry-After TLV is defined that the TEAP server uses to specify a retry interval in seconds. New error codes are defined to handle these two alternate retry scenarios.

- o The TEAP tunnel remains up: The client is instructed to resend the PKCS#10 request after a retry interval but inside the same TEAP tunnel. The TEAP server returns a Retry-After TLV to the client, and returns an Error TLV with a new code in the 1000-1999 range.

- o The TEAP tunnel is torn down: The client is instructed to establish a new TEAP connection and TEAP tunnel after a retry interval, and resend the PKCS#10 request indside the new tunnel. The TEAP server returns a Retry-After TLV to the client, and returns an Error TLV with a new code in the 2000-2999 range.

5. Peer Identity

EAP [RFC3748] recommends that "the Identity Response be used primarily for routing purposes and selecting which EAP method to use". NAI [RFC7542] recommends ommitting the username part of an NAI in order to support username privacy, where appropriate.

A device that has not been bootstrapped at all SHOULD send an identity of teap-bootstrap@TBD1. Otherwise, a device SHOULD send its configured NAI.

The TEAP server may specify an NAI that it wishes the device to use. For example, the server may want a bootstrapped device to use an NAI of "abc123@example.com", or simply an NAI of "@example.com". This could be desirable in order to facilitate roaming scenarios. The server can do this by sending the device an NAI TLV inside the TEAP tunnel.

If the server specifies an NAI TLV, and the device handles the TLV, the device MAY use the specified NAI in all subsequent EAP authentication flows. If the device is not willing to handle the NAI TLV, it MUST reply with an Error TLV.

Authentication servers implementing this specification MAY reply with an Error TLV to any unrecognized NAI, or MAY attempt to bootstrap the device, regardless of the NAI. A device receving an Error from the server MAY attempt a new session without the NAI in order to bootstrap.

6. Channel and Crypto Binding

As the TEAP BRSKI flow does not define or require an inner EAP method, there is no explicit need for exchange of Channel-Binding TLVs between the device and the TEAP server.

The TEAP BRSKI TLVs are expected to occur at the beginning of the TEAP Phase 2 and MUST occur before the final Crypto-Binding TLV. This draft does not exclude the possibility of having other EAP methods occur following the TEAP BRSKI TLVs and as such, the Crypto-Binding TLV process rules as defined in [RFC7170] apply.

7. Protocol Flows

This section outlines protocol flows that map to the three server policy options described in section Section 4.1. The protocol flows illustrate a TLS1.2 exchange. Pertinent notes are outlined in the protocol flows.

7.1. TEAP Server Grants Access

In this flow, the server grants access as server policy allows the client to access the network based on the identity certificate that the client presented. This means that either (i) the client has previously completed BRSKI and has presented a valid LDevID or (ii) the client presents an IDevID and network policy allows access based purely on IDevID.

```
,------.              ,----------.
|Client|              |TEAPServer|
`--+---'              `----+-----'
   |     EAP-Request/      |
   |     Identity          |
   | <----------------------
   |                       |
   |     EAP-Response/     |
   |     Type=Identity     |
```

```
             |  --------------------->
             |                       |
             |    EAP-Request/       |
             |    Type=TEAP,         |
             |    TEAP Start,        |
             |    Authority-ID TLV   |
             |  <---------------------
             |                       |
             |    EAP-Response/      |
             |    Type=TEAP,         |
             |    TLS(ClientHello)   |
             |  --------------------->
             |                       |
             |   EAP-Request/        |
      ,---!. |   Type=TEAP,          |
      |(1)|_\|   TLS(ServerHello,    |
      |(2)  ||   ServerKeyExchange,  |
      `-----'|   ServerHelloDone)    |
             |  <---------------------
             |                       |
             |   EAP-Response/       |
             |   Type=TEAP,          |
      ,---!. |   ClientKeyExchange,  |
      |(3)|_\|   CertificateVerify,  |
      `-----'|   ChangeCipherSpec,   |
             |   Finished)           |
             |  --------------------->
             |                       |
             |  EAP-Request/         |
             |  Type=TEAP,           |
             |  TLS(ChangeCipherSpec,|
             |  Finished),           |
             |  {Crypto-Binding TLV, |
             |  Result TLV=Success}  |
             |  <---------------------
             |                       |
             |  EAP-Response/        |
             |  Type=TEAP,           |
             |  {Crypto-Binding TLV, |
             |  Result TLV=Success}  |
             |  --------------------->
             |                       |
             |       EAP-Success     |
             |  <---------------------
           ,--+---.             ,----+-----.
           |Client|             |TEAPServer|
           `------'             `----------'
```

Figure 1: TEAP Server Grants Access

Notes:

(1) If the client has completed the BRSKI flow and has locally significant trust anchors, it must validate the Certificate received from the server. If the client has not yet completed the BRSKI flow, then it provisionally accepts the server Certificate and must validate it later once BRSKI is complete.

(2) The server may include certificate_authorities field in the CertificateRequest message in order to restrict the identity certificates that the device is allowed present.

(3) The device will present its LDevID, if it has one, in preference to its IDevID, if allowed by server policy.

7.2. TEAP Server Instructs Client to Perform BRSKI Flow

In this two part flow, the server instructs the client to perform a BRSKI flow by exchanging TLVs once the outer TLS tunnel is established. After that, enrollment takes place.

In the first part of the flow, the MASA is depicted on the right.

```
         ,------.                           ,----------.          ,----.
         |Client|                           |TEAPServer|          |MASA|
         `--+---'                           `----+-----'          `-+--'
            |            EAP-Request/            |                  |
            |            Type=Identity           |                  |
            | <-----------------------------------                  |
            |                                    |                  |
            |            EAP-Response/           |                  |
            |            Type=Identity           |                  |
            | ----------------------------------->                  |
            |                                    |                  |
            |          EAP-Request/              |                  |
     ,---!. |          Type=TEAP,                |                  |
     |(1)|_\|          TEAP Start,               |                  |
     `-----'|          Authority-ID TLV          |                  |
            | <-----------------------------------                  |
            |                                    |                  |
            |          EAP-Response/             |                  |
            |          Type=TEAP,                |                  |
            |          TLS(ClientHello)          |                  |
            | ----------------------------------->                  |
            |                                    |                  |
            |         EAP-Request/               |                  |
```

```
            |         Type=TEAP,                 |                  |
            |         TLS(ServerHello,           |                  |
            |         Certificate,               |                  |
            |         ServerKeyExchange,         |                  |
            |         CertificateRequest,        |                  |
            |         ServerHelloDone)           |                  |
            | <-----------------------------------                  |
            |                                    |                  |
            |         EAP-Response/              |                  |
            |         Type=TEAP,                 |                  |
            |         TLS(Certificate            |                  |
            |         ClientKeyExchange,         |                  |
            |         CertificateVerify,         |                  |
            |         ChangeCipherSpec,          |                  |
            |         Finished)                  |                  |
            | ----------------------------------->                  |
            |                                    |                  |
            |        EAP-Request/                |                  |
            |        Type=TEAP,                  |                  |
            |        TLS(ChangeCipherSpec,       |                  |
            |        Finished),                  |                  |
            |        {Crypto-Binding TLV,        |                  |
            |        Result TLV=Success}         |                  |
            | <-----------------------------------                  |
            |                                    |                  |
            |        EAP-Response/               |                  |
            |        ,Type=TEAP,                 |                  |
            |        {Crypto-Binding TLV,        |                  |
            |        Result TLV=Success}         |                  |
            | ----------------------------------->                  |
            |                                    |                  |
     ,-------------------------------------------------!.           |
     |At this stage the outer TLS tunnel is established|_\          |
     |The following message exchanges are for BRSKI      |          |
     `---------------------------------------------------'          |
            |      EAP-Request/                  |                  |
            |      Type=TEAP,                    |                  |
            |      {Request-Action TLV:          |                  |
            |      Status=Failure,               |                  |
     ,---!. |      Action=Process-TLV,           |                  |
     |(2)|_\      TLV=Request-Voucher,           |                  |
     `-----'|      TLV=Trusted-Server-Root,      |                  |
            |      TLV=CSR-Attributes,           |                  |
            |      TLV=PKCS#10}                  |                  |
            | <-----------------------------------                  |
            |                                    |                  |
            |        EAP-Response/               |                  |
     ,---!. |        Type=TEAP,                  |                  |
```

```
     |(3)|_\|        {Request-Voucher TLV}       |                  |
     `-----'| ----------------------------------->                  |
            |                                    |                  |
            |                                    |  RequestVoucher  |
            |                                    | ----------------->
            |                                    |                  |
            |                                    |      Voucher     |
            |                                    | <-----------------
            |                                    |                  |
            |            EAP-Request/            |                  |
     ,---!. |            Type=TEAP,              |                  |
     |(4)|_\|            {Voucher TLV}           |                  |
     `-----'| <-----------------------------------                  |
            |                                    |                  |
            | EAP-Response/                      |                  |
            | Type=TEAP,{Trusted-Server-Root TLV}|                  |
            | ----------------------------------->                  |
            |                                    |                  |
            |      EAP-Request/                  |                  |
     ,---!. |      Type=TEAP,                    |                  |
     |(5)|_\|      {Trusted-Server-Root TLV}     |                  |
     `-----'| <-----------------------------------                  |
            |                                    |                  |
            |   EAP-Response/                    |                  |
            |   Type=TEAP,{CSR-Attributes TLV}   |                  |
            | ----------------------------------->                  |
            |                                    |                  |
            |        EAP-Request/                |                  |
            |        Type=TEAP,                  |                  |
            |        {CSR-Attributes TLV}        |                  |
            | <-----------------------------------                  |
         ,--+---.                           ,----+-----.          ,-+--.
         |Client|                           |TEAPServer|          |MASA|
         `------'                           `----------'          `----'
```

Figure 2: TEAP Server Instructs Client to Perform BRSKI Flow

The second part of the flow depicts the CA on the right.

```
        ,------.            ,----------.          ,--.
        |Client|            |TEAPServer|          |CA|
        `--+---'            `----+-----'          `+-'
           |    EAP-Response/    |                 |
           |    Type=TEAP        |                 |
           |    {PKCS#10 TLV}    |                 |
           | -------------------->                 |
           |                     |                 |
           |                     |    PKCS#10      |
           |                     | ---------------->
           |                     |                 |
           |                     |     PKCS#7      |
           |                     | <----------------
           |                     |                 |
           | EAP-Request/        |                 |
           | Type=TEAP,          |                 |
           | {PKCS#7 TLV,        |                 |
           | Result TLV=Success} |                 |
           | <--------------------                 |
           |                     |                 |
           | Eap-Response/       |                 |
           | Type=TEAP           |                 |
           | {Result TLV=Success}|                 |
           | -------------------->                 |
           |                     |                 |
           |     EAP-Success     |                 |
           | <--------------------                 |
        ,--+---.            ,----+-----.          ,+-.
        |Client|            |TEAPServer|          |CA|
        `------'            `----------'          `--'
```

Figure 3: Enrollment after BRSKI Flow

Notes:

(1) If the client has not yet completed the BRSKI flow, then it provisionally accepts the server certificate and must validate it later once BRSKI is complete. The server validates the client certificate using its trust anchor database.

(2) The server instructs the client to start the BRSKI flow by sending a Request-Action TLV that includes a BRSKI-RequestVoucher TLV. The server also instructs the client to request trust anchors, to request CSR Attrites, and to initiate a PKCS certificate enrolment. As outlined in [RFC7170], the Request-Action TLV is sent after the Crypto-Binding TLV and Result TLV exchange.

(3) The client includes the certificate it received from the server in the RequestVoucher message.

(4) Once the client receives and validates the voucher signed by the MASA, it must verify the certificate it previously received from the server.

(5) As outlined in [RFC7170], the Trusted-Server-Root TLV is exchanged after the Crypto-Binding TLV exchange, and after the client has used the Voucher to authenticate the TEAP server identity. This is equivalent to section [todo] from [I-D.ietf-anima-bootstrapping-keyinfra].

(6) There is no need for an additional Crypto-Binding TLV exchange as there is no inner EAP method. All BRSKI exchanges are simply TLVs exchanged inside the outer TLS tunnel.

7.3. TEAP Server Instructs Client to Reenroll

In this flow, the server instructs the client to reenroll and get a new LDevID by exchanging TLVs once the outer TLS tunnel is established.

```
         ,------.                      ,----------.          ,--.
         |Client|                      |TEAPServer|          |CA|
         `--+---'                      `----+-----'          `+-'
            |          EAP-Request/         |                 |
            |          Identity             |                 |
            | <------------------------------                 |
            |                               |                 |
            |         EAP-Response/         |                 |
            |         Type=Identity         |                 |
            | ------------------------------>                 |
            |                               |                 |
            |        EAP-Request/           |                 |
            |        Type=TEAP,             |                 |
            |        TEAP Start,            |                 |
            |        Authority-ID TLV       |                 |
            | <------------------------------                 |
            |                               |                 |
            |        EAP-Response/          |                 |
            |        Type=TEAP,             |                 |
            |        TLS(ClientHello)       |                 |
            | ------------------------------>                 |
            |                               |                 |
            |       EAP-Request/            |                 |
            |       Type=TEAP,              |                 |
            |       TLS(ServerHello,        |                 |
```

```
            |       ServerKeyExchange,      |                 |
            |       ServerHelloDone)        |                 |
            | <------------------------------                 |
            |                               |                 |
            |       EAP-Response/           |                 |
            |       Type=TEAP,              |                 |
            |       ClientKeyExchange,      |                 |
            |       CertificateVerify,      |                 |
            |       ChangeCipherSpec,       |                 |
            |       Finished)               |                 |
            | ------------------------------>                 |
            |                               |                 |
            |     EAP-Request/              |                 |
            |     Type=TEAP,                |                 |
            |     TLS(ChangeCipherSpec,     |                 |
            |     Finished),                |                 |
            |     {Crypto-Binding TLV,      |                 |
            |     Result TLV=Success}       |                 |
            | <------------------------------                 |
            |                               |                 |
            |      EAP-Response/            |                 |
            |      Type=TEAP,               |                 |
            |      {Crypto-Binding TLV,     |                 |
            |      Result TLV=Success}      |                 |
            | ------------------------------>                 |
            |                               |                 |
            | EAP-Request/                  |                 |
            | Type=TEAP,{Request-Action TLV:|                 |
     ,---!. | Status=Failure,               |                 |
     |(1)|_\| Action=Process-TLV,           |                 |
     `-----'| TLV=PKCS#10}                  |                 |
            | <------------------------------                 |
            |                               |                 |
            |        EAP-Response/          |                 |
            |        Type=TEAP              |                 |
            |        {PKCS#10 TLV}          |                 |
            | ------------------------------>                 |
            |                               |                 |
            |                               |     PKCS#10     |
            |                               | ---------------->
            |                               |                 |
            |                               |      PKCS#7     |
            |                               | <----------------
            |                               |                 |
            |      EAP-Request/             |                 |
            |      Type=TEAP,               |                 |
            |      {PKCS#7 TLV,             |                 |
            |      Result TLV=Success}      |                 |
```

```
           | <-------------------------------               |
           |                               |                |
           |      Eap-Response/            |                |
           |      Type=TEAP                |                |
           |      {Result TLV=Success}     |                |
           | ------------------------------>                |
           |                               |                |
           |          EAP-Success          |                |
           | <-------------------------------               |
           |                               |                |
           |          EAP-Success          |                |
           | <-------------------------------               |
        ,--+---.                      ,----+-----.        ,+-.
        |Client|                      |TEAPServer|        |CA|
        `------'                      `----------'        `--'
```

Figure 4: TEAP Server Instructs Client to Reenroll

(1) The server instructs the client to reenroll by sending a Request-Action TLV that includes a PKCS#10 TLV.

7.4. Out of Band Reenroll

This section shows how the device does a reenroll to refresh its LDEvID directly against the registrar outside the context of the TEAP tunnel.

8. TEAP TLV Formats

8.1. New TLVs

This document defines 5 new TEAP TLVs. The following table indicates whether the TLVs can be included in Request messages from TEAP server to device, or Response messages from device to TEAP server.

TLV	Message
BRSKI-VoucherRequest	Response
BRSKI-Voucher	Request
CSR-Attributes	Response
Retry-After	Response
NAI-Identity	Request

These new TLVs are detailed in this section.

8.1.1. BRSKI-RequestVoucher TLV

This TLV is used by the server as part of a Request-Action TLV to request from the peer that it initiate a voucher request. When used in this fashion, the length of this TLV will be set to zero. The Status field of the Request-Action TLV MUST be set to Failure.

It is also used by the peer to initiate the voucher request. When used in this fashion, the length of the TLV will be set to that of the voucher request, as encoded and described in Section 3.3 in [I-D.ietf-anima-bootstrapping-keyinfra].

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|M|R| TLV=TBD1-VoucherRequest   |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              Value...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The M and R bits are always expected to be set to 0.

The server is expected to forward the voucher request to the MASA, and then return a voucher in a BRSKI-Voucher TLV as described below. If it is unable to do so, it returns an TEAP Error TLV with one of the defined errors or the following:

```
TBD2-MASA-Notavailable  MASA unavailable
TBD3-MASA-Refused       MASA refuses to sign the voucher
```

The peer terminates the TEAP connection, but may retry at some later point. The backoff mechanism for such retries should be appropriate for the device. Retries MUST occur no more frequently than once every two (TBD) minutes.

8.1.2. BRSKI-Voucher TLV

This TLV is transmitted from the server to the peer. It contains a signed voucher, as describe in [RFC8366].

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|M|R| TLV=TBD4-Voucher          |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              Value...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Upon receiving this TLV the peer will validate the signature of the voucher, using its pre-installed manufacturer trust anchor (LDevID). It MUST also validate the certificate used by the server to establish the TLS connection.

If successful, it installs the new trust anchor contained in the voucher.

Otherwise, the peer transmits an TEAP error TLV with one of the following error messages:

```
TBD5-Invalid-Signature  The signature on the voucher is invalid
TBD6-Invalid-Voucher    The form or content of the voucher is invalid
TBD7-Invalid-TLS-Signer The certificate used for the TLS connection
                        could not be validated.
```

8.1.3. CSR-Attributes TLV

The server SHALL transmit this TLV to the peer, either along with the BRSKI-Voucher TLV or at any time earlier in a communication. The peer shall include attributes required by the server in any following CSR. The value of this TLV is the base64 encoding described in Section 4.5.2 of [RFC7030].

The TEAP server MAY use this TLV to specify the subject identity information to include in Subject or Subjecet Alternate Name fields in any following CSR.

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|M|R| TLV=TBD8-CSR-Attributes |         length                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Value...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Again, the M and R values are set to 0. In the case where the client is unable to provide the requested attributes, an TEAP-Error is returned as follows:

TBD9-CSR-Attribute-Fail Unable to supply the requested attributes.

8.1.4. Retry-After TLV

The server MUST transmit this TLV to the peer when repling to a PKCS#10 TLV request from the peer where the server is willing to fulfill the request and issue a certificate via a PKCS#7 response, but is unable to fulfill the request immediately. This TLV is used to tell the peer the mimimum lenght of time it MUST wait before resending the PKCS#10 request. The value of this TLV is the time in seconds that the peer MUST wait before resending the PKCS#10 request. The peer MUST resend the exact same PKCS#10 request.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|M|R| TLV=TBD10-Retry-After     |          length               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Value...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Again, the M and R values are set to 0.

8.1.5. NAI TLV

The server may use this TLV to provision a realm-specific NAI on the device.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|M|R| TLV=TBD10-NAI             |          length               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Value...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Again, the M and R values are set to 0.

8.2. Existing TEAP TLV Specifications

This section documents allowed usage of existing TEAP TLVs. The definition of the TLV is not changed, however clarifications on allowed values for the TLV fields is documented.

8.2.1. PKCS#10 TLV

[RFC7170] defines the PKCS#10 TLV as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|M|R|         TLV Type          |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           PKCS#10 Data...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-++-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

[RFC7170] does not explicitly allow a Length value of zero.

A Length value of zero is allowed for this TLV when the TEAP server sends a Request-Action TLV with a child PKCS#10 TLV to the client. In this scenario, there is no PKCS#10 Data included in the TLV. Clients MUST NOT send a zero length PKCS#10 TLV to the server.

8.3. TLV Rules

BRSKI TLVs can only be transported inside the TLS tunnel. The following table provides a guide to which TLVs may be encapsulated in which kind of packets, and in what quantity. The messages are as follows: Request is a TEAP Request, Response is a TEAP Response, Success is a message containing a successful Result TLV, and Failure is a message containing a failed Result TLV.

The following define the meaning of the table entries in the sections below:

0 This TLV MUST NOT be present in the message.

0+ Zero or more instances of this TLV MAY be present in the message.

0-1 Zero or one instance of this TLV MAY be present in the message.

1 Exactly one instance of this TLV MUST be present in the message.

Request Response Success Failure TLVs 0 0-1 0 0 BRSKI-VoucherRequest 0-1 0 0 0 BRSKI-Voucher 0 0-1 0 0 CSR-Attributes

9. Fragmentation

TEAP is expected to provide fragmentation support. Thus EAP-TEAP-BRSKI does not specifically provide any, as it is only expected to be used as an inner method to TEAP.

10. IANA Considerations

The IANA is requested to add entries into the following tables:

The following new TEAP TLVs are defined:

```
TBD1-VoucherRequest   Described in this document.
TBD4-Voucher          Described in this document.
TBD8-CSR-Attributes   Described in this document.
TBD10-Retry-After Described in this document.
```

The following TEAP Error Codes are defined, with their meanings listed here and in previous sections:

```
TBD2-MASA-Notavailable  MASA unavailable
TBD3-MASA-Refused       MASA refuses to sign the voucher
TBD5-Invalid-Signature  The signature on the voucher is invalid
TBD6-Invalid-Voucher    The form or content of the voucher is invalid
TBD7-Invalid-TLS-Signer The certificate used for the TLS connection
                        could not be validated.
TBD9-CSR-Attribute-Fail Unable to supply the requested attributes.
TBD11-Retry-PKCS#10     Retry PKCS#10 Request (1000 range code)
TBD12-Retry-PKCS#10     Retry PKCS#10 Request (2000 range code)
TBD13-NAI-Rejected      The device will not use the indicated
                        NAI (1000 range code)
```

[[TODO: is there a registry of NAIs that map to TEAP methods? e.g. @eap-teap.net is reserved to indicate the peer wants to use TEAP method]]

11. Security Considerations

BRSKI [I-D.ietf-anima-bootstrapping-keyinfra] provides a zero touch way for devices to enroll in a certification authority (CA). It assumes the device has IP connectivity. For networks that will not grant IP connectivitiy before authenticating (with a local credential) this poses a Catch-22- can't get on the network without a credential and can't get a credential without getting on the network.

This protocol provides a way for BRSKI to be in an EAP method which allows the BRSKI conversation to happen as part of EAP authentication and prior to obtaining IP connectivity.

The security considerations of [I-D.ietf-anima-bootstrapping-keyinfra] apply to this protocol. Running BRSKI through EAP introduces some additional areas of concern though.

11.1. Issues with Provisionally Authenticated TEAP

This protocol establishes an unauthenticated TLS connection and passes data through it. Provided that the only messages passed in this state are self-protected BRSKI messages this does not present a problem. Passing any other messages or TLVs prior to authentication of the provisional TLS connection could potentially introduce security issues.

While the TLS connection is unauthenticated, it must still be validated to the fullest extent possible. It is critical that the device and the TEAP server perform all steps in TLS- checking the validity of the presented certificate, validating the signature using the public key of the certificate, etc- except ensuring the trust of the presented certificate.

11.2. Attack Against Discovery

The device discovery technique specified in this protocol is the standard EAP server discovery process. Since it is trivial to set up an 802.11 wireless access point and advertise any network, an attacker can impersonate a legitimate wireless network and attract unprovisioned pledges. Given that an unprovisioned device will not know the legitimate network to connect to, it will probably attempt the first network it finds, making the attack that much easier. This allows for a "rogue registrar" to provision and take control of the device.

If the MASA verifies ownership prior to issuance of a voucher, this attack can be thwarted. But if the MASA is in reduced security mode and does not verify ownership this attack cannot be prevented. Registrars SHOULD use the audit log of a MASA when deploying newly purchased equipment in order to mitigate this attack.

Another way to mitigate this attack is through normal "rogue AP" detection and prevention.

11.3. TEAP Server as Registration Authority

If the TEAP server is logically separate from the Certification Authority (CA) (see Section 2) it will be acting as a Registration Authority (RA) when it obtains the PKCS#10 TLV and replies with a PKCS#7 TLV (see [RFC7170], Sections 4.2.16 and 4.2.17, respectively). The assurance a RA makes to a CA is that the public key in the presented CSR is bound to an authenticated identity in way that will assure non-repudiation.

To make such an assurance, the TEAP server MUST authenticate the provisional TLS connection with the device by validating the voucher response received from the MASA. In addition, it is RECOMMENDED that the TEAP server indicate that proof-of-possession (see [RFC7170], Section 3.8.2) is required by including the challengePassword OID in the CSR Attributes TLV.

11.4. Trust of Registrar

The device accepts a trusted server (CA) certificate and installs it in its trust anchor database during step 5 (see Section 3.2). This can happen only after the provisional TLS connection has been authenticated using the voucher and the Crypto-Binding TLV has been validated.

12. Acknowledgments

The authors would like to thank Brian Weis for his assistance, and Alan Dakok for improving language consistency. In addition, with ruthlessly "borrowed" the concept around NAI handling from Tuomas Aura and Mohit Sethi.

13. References

13.1. Normative References

[I-D.ietf-anima-bootstrapping-keyinfra]
Pritikin, M., Richardson, M., Eckert, T., Behringer, M., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructures (BRSKI)", draft-ietf-anima-bootstrapping-keyinfra-29 (work in progress), October 2019.

[IEEE8021AR]
Institute for Electrical and Electronics Engineers, "Secure Device Identity", 1998.

[IEEE8021X]
Institute for Electrical and Electronics Engineers, "IEEE Standard for Local and metropolitan area networks--Port-Based Network Access Control", 2010.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, Ed., "Extensible Authentication Protocol (EAP)", RFC 3748, DOI 10.17487/RFC3748, June 2004, <https://www.rfc-editor.org/info/rfc3748>.

[RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed., "Enrollment over Secure Transport", RFC 7030, DOI 10.17487/RFC7030, October 2013, <https://www.rfc-editor.org/info/rfc7030>.

[RFC7170] Zhou, H., Cam-Winget, N., Salowey, J., and S. Hanna, "Tunnel Extensible Authentication Protocol (TEAP) Version 1", RFC 7170, DOI 10.17487/RFC7170, May 2014, <https://www.rfc-editor.org/info/rfc7170>.

[RFC8366] Watsen, K., Richardson, M., Pritikin, M., and T. Eckert, "A Voucher Artifact for Bootstrapping Protocols", RFC 8366, DOI 10.17487/RFC8366, May 2018, <https://www.rfc-editor.org/info/rfc8366>.

13.2. Informative References

[RFC7542] DeKok, A., "The Network Access Identifier", RFC 7542, DOI 10.17487/RFC7542, May 2015, <https://www.rfc-editor.org/info/rfc7542>.

Appendix A. Changes from Earlier Versions

Draft -03: * Merge EAP server and Registrar * Security considerations * References improvements * Add Dan Harkins as co-author

Draft -02: * Flow corrections

Draft -01: * Add packet descriptions, IANA considerations, smooth out language.

Draft -00:

- o Initial revision

Authors' Addresses

Eliot Lear
Cisco Systems
Richtistrasse 7
Wallisellen CH-8304
Switzerland

Phone: +41 44 878 9200
Email: lear@cisco.com

Owen Friel
Cisco Systems
170 W. Tasman Dr.
San Jose, CA 95134
United States

Email: ofriel@cisco.com

Nancy Cam-Winget
Cisco Systems
170 W. Tasman Dr.
San Jose, CA 95134
United States

Email: ncamwing@cisco.com

Dan Harkins
HP Enterprise
3333 Scott Boulevard
Santa Clara, CA 95054
United States

Email: dharkins@arubanetworks.com

Network Working Group E. Lear
Internet-Draft J. Henry
Intended status: Experimental Cisco Systems
Expires: January 9, 2020 July 08, 2019

Bandwidth Profiling Extensions for MUD
draft-lear-opsawg-mud-bw-profile-01

Abstract

Manufacturer Usage Descriptions (MUD) are a means by which devices can establish expectations about how they are intended to behave, and how the network should treat them. Earlier work focused on access control. This draft specifies a means by which manufacturers can express to deployments what form of bandwidth profile devices are expected to have with respect to specific services.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Devices connecting to networks will often exhibit certain nominal behaviors that can be described. In addition, sometimes device require particular network behaviors such as appropriate quality-of-service treatment. Manufacturer Usage Descriptions [RFC8520] discuss how to characterize access control requirements, for instance. As just mentioned, access control requirements are not the only requirements device manufacturers may wish to specify. This memo defines an extension to the MUD YANG model by which manufacturers can characterize the traffic exchanged with a Thing, and specify how much bandwidth is required by a device or may be expected of a device over some period of time for each given service it uses.

Network deployments may use this information in two ways:

- o Provisioning of bandwidth based on device requirements;
- o Facilitating proper traffic characterization and marking by the network infrastructure
- o Policing of devices to not permit them to exceed design requirements. In particular, a device that is transmitting a DSCP value that exceeds the expected value, or that manifests unusual transmission patterns, should be viewed with great suspicion.

The basis of the model is that services may be identified by access-lists, and that each service can then be assigned an attendant bandwidth expectation in terms of either bits-per-second or packets-per-second. In addition, a DSCP marking can be specified.

When a service is identified by access lists, each access list is appended to the existing access list entries. N.B., as a reminder, acl names in MUD files are scoped solely to those files, and may conflict with acl names in _other_ MUD files.

1.1. Envisioned Uses

A luminaire may require a few packets per minute of a predictable payload size (e.g. keepalives), and may expect that traffic to be sent in the background, as one or more keepalive packet loss would not impede the luminaire functions. Additionally, when a virtual 'light switch' changes its state, a burst of 3 to 4 packets over a well-defined port are expected, with a QoS marking of OAM. Last, occasional firmware updates may bring an exchange of a few kilobytes marked as best effort.

A smoke detector may require at most 1 packet per second at best effort (keepalive), except when there is a problem, at which point it may send a frame upstream to a specific port and of a specified payload size, with a DSCP marking of EF.

A coffee maker may be designed never set DSCP to anything other than AF13 (even when it's empty, perish the thought), nor may it ever use more than 5 packets of 120 bytes payload per minute, even if it has a fault.

A different coffee maker may be designed to set DSCP to EF if the it has caught fire.

1.2. Limitations

Not every device can be easily profiled. Not every service on every device may be easily profiled. A manufacturer may use this extension to describe those services that _are_ easily profile, and omit services that the device offers or uses that are not easily profiled. The local deployment is cautioned not to assume that a service not profiled is in some way anomalous, even when other services are.

1.3. What devices would use this extension?

The MUD manager remains a key component of this system. To begin with, it is the component that retrieves the MUD file, and would identify the extension. From that point, different implementation decisions can be made. For instance, the MUD manager or associated infrastructure can retain the mapping between devices and MUD-URLs. A dispatch function could be implemented wherever that mapping is housed, such that either enforcement or monitoring functions can be invoked. Enforcement functions would almost certainly begin with some form of telemetry on access switches, routers or firewalls. That same telemetry might be exported to an IPFIX analyzer [RFC7011] that might report anomalies.

2. The ietf-mud-bw-profile model extension

To extend MUD the "qos" extension is added as an element to the "extensions" node when a MUD file is generated.

The model augmentation appears as follows:

```
module: ietf-mud-bw-profile
  augment /mud:mud/mud:to-device-policy:
    +--rw bw-params
       +--rw service* [name]
          +--rw name         string
          +--rw timeframe    uint32
          +--rw pps?         uint32
          +--rw bps?         uint64
          +--rw dscp?        inet:dscp
          +--rw aclname?     -> /acl:acls/acl/name
  augment /mud:mud/mud:from-device-policy:
    +--rw bw-params
       +--rw service* [name]
          +--rw name         string
          +--rw timeframe    uint32
          +--rw pps?         uint32
          +--rw bps?         uint64
          +--rw dscp?        inet:dscp
          +--rw aclname?     -> /acl:acls/acl/name
```

2.1. The mud-qos YANG model

```
 <CODE BEGINS>file "ietf-mud-bw-profile@2019-07-08.yang"
 module ietf-mud-bw-profile {
   yang-version 1.1;
   namespace "urn:ietf:params:xml:ns:yang:ietf-mud-bw-profile";
   prefix mud-qos;

   import ietf-access-control-list {
     prefix acl;
   }
   import ietf-inet-types {
```

```
     prefix inet;
   }
   import ietf-mud {
     prefix mud;
   }

   organization
     "IETF OPSAWG (Ops Area) Working Group";
   contact
     "WG Web: http://tools.ietf.org/wg/opsawg/
      WG List: opsawg@ietf.org
      Author: Eliot Lear
      lear@cisco.com
      Author: Jerome Henry
      jerhenry@cisco.com
     ";
   description

     "This YANG module augments the ietf-mud model to provide the
      network with some understanding as to the QoS requirements and
      anticipated behavior of a device.

      The to-device-policy and from-device-policy containers are
      augmented with one additional container, which expresses how many
      packets per second a device is expected to transmit, how much
      bandwidth it is expected to use, and what QoS is required, and
      how much bandwidth is to be expected to be prioritized.  An
      access-list is further specified to indicate how QoS should be
      marked on ingress and egress.

      Copyright (c) 2016,2017,2018 IETF Trust and the persons
      identified as the document authors.  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).
      This version of this YANG module is part of RFC XXXX; see
      the RFC itself for full legal notices.";

   revision 2019-07-08 {
     description
       "Initial proposed standard.";
     reference "RFC XXXX: Bandwidth Descriptions for MUD Specification";
   }

   grouping mud-bw-params {
```

```
     description
       "QoS and Bandwidth additions for MUD";
     container bw-params {
       description
         "Expected Bandwidth to/from device";
       list service {
         key "name";
         description
           "a list of services that are being described.";
         leaf name {
           type string;
           description
             "Service Name";
         }
         leaf timeframe {
           type uint32;
           mandatory true;
           description
             "the period of time in seconds one
              expects a service to burst at described rates";
         }
         leaf pps {
           type uint32;
           description
             "number of packets per second to be expected.";
         }
         leaf bps {
           type uint64;
           description
             "number of bits per second to be expected.";
         }
         leaf dscp {
           type inet:dscp;
           description
             "The DSCP that packets for this service should
              treated with.  N.B., just because the manufacturer
              wants this, doesn't mean it will get it.  However,
              manufacturers who do set the DSCP value in their
              packets SHOULD indicate that in this description.

              This field differs from the dscp field in the matches
              portion of the access-list in that here the field is
              populated when the manufacturer states what the nominal
              value of the DSCP field MAY be, and how much bandwidth
              can be used when it is set.  Note that it is possible
              that the same service may use multiple DSCP values,
              depending on the circumstances.  In this case, service
              entry MUST be made.";
```

```
         }
         leaf aclname {
           type leafref {
             path "/acl:acls/acl:acl/acl:name";
           }
           description
             "The name of the ACL that will match packets
              for a given service.";
         }
       }
     }
   }

   augment "/mud:mud/mud:to-device-policy" {
     description
       "add inbound QoS parameters";
     uses mud-bw-params;
   }
   augment "/mud:mud/mud:from-device-policy" {
     description
       "add outbound QoS parameters";
     uses mud-bw-params;
   }
 }
 <CODE ENDS>
```

3. Examples

TBD

4. Security Considerations

4.1. Manufacturer Attempts to Exhaust Available Bandwidth

An attacking manufacturer claims a device would require substantial bandwidth or QoS for use. This attack would be effected when a device is installed into a local deployment and its MUD file interpreted. The impact of a device demanding excessive bandwidth could be overprovisioning of the network or denial of service to other uses.

This attack is remediated by a human being reviewing the bandwidth consumption projections suggested by the MUD file when they are in some way beyond the norm for any device being installed.

4.2. Device lies about what it is to get more bandwidth

If the device is emitting a MUD-URL via insecure, it is possible for an attacker to modify it. Devices emitting such URLs should already receive additional scrutiny from administrators as they are onboarded. This mechanism SHOULD NOT be used to admit devices into privileged queues without them having been securely admitted to the network, through means such as IEEE 802.1X.

5. IANA Considerations

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

Extension Name: MUD
Standard reference: This document

6. References

6.1. Normative References

[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>.

6.2. Informative References

[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>.

Appendix A. Changes from Earlier Versions

Draft -01:

- Very modest changes.

Draft -00:

- Initial revision

Authors' Addresses

Eliot Lear
Cisco Systems
Richtistrasse 7
Wallisellen CH-8304
Switzerland

Phone: +41 44 878 9200
Email: lear@cisco.com

Jerome Henry
Cisco Systems
170 W. Tasman Dr.
San Jose, CA 95134
United States

Email: ofriel@cisco.com

OPSAWG
Internet-Draft
Intended status: Informational
Expires: April 8, 2021

H. Song
Futurewei
F. Qin
China Mobile
H. Chen
China Telecom
J. Jin
LG U+
J. Shin
SK Telecom

October 5, 2020

In-situ Flow Information Telemetry
draft-song-opsawg-ifit-framework-13

Abstract

As networks increase in scale and network operations become more sophisticated, traditional Operation, Administration and Maintenance (OAM) methods, which include proactive and reactive techniques, running in active and passive modes, are no longer sufficient to meet the monitoring and measurement requirements. Emerging on-path telemetry techniques which provide high-precision flow insight and real-time issue notification are required to ensure suitable quality of experience for users and applications, and identify faults or network deficiencies before they become critical.

This document outlines a high-level framework to provide an operational environment that utilizes existing and emerging on-path telemetry techniques to enable the collection and correlation of performance information from the network. The framework identifies the components that are needed to coordinate the existing protocol tools and telemetry mechanisms, and addresses key deployment challenges for flow-oriented on-path telemetry techniques, especially in carrier networks.

The framework is informational and intended to guide system designers attempting to use the referenced techniques as well as to motivate further work to enhance the ecosystem .

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Efficient network operation increasingly relies on high-quality data-plane telemetry to provide the necessary visibility. Traditional Operation, Administration and Maintenance (OAM) methods, which include proactive and reactive techniques, running both active and passive modes, are no longer sufficient to meet the monitoring and measurement requirements. The complexity of today's networks and service quality requirements demand new high-precision and real-time techniques.

The ability to expedite network failure detection, fault localization, and recovery mechanisms, particularly in the case of soft failures or path degradation is expected, without causing service disruption. Networks also become application-aware. Application-aware networking is an industry term used to describe the capacity of a network to maintain current information about users and application connections which may be used to optimize the network resource usage and improve the quality of service.

The emerging on-path telemetry techniques can provide high-precision flow insight and real-time network issue notification (e.g., jitter, latency, packet loss, significant bit error variations, and unequal load-balancing). On-path telemetry refers to the data-plane telemetry techniques that directly tap and measure network traffic by embedding instructions or metadata into user packets. The data provided by on-path telemetry are especially useful for SLA compliance, user experience enhancement, service path enforcement, fault diagnosis, and network resource optimization. It is essential to recognize that existing work on this topic includes a variety of on-path telemetry techniques, including In-situ OAM (IOAM) [I-D.ietf-ippm-ioam-data], Postcard-based Telemetry (PBT) [I-D.song-ippm-postcard-based-telemetry], Enhanced Alternate Marking (EAM) [I-D.zhou-ippm-enhanced-alternate-marking], and Hybrid Two Steps (HTS) [I-D.mirsky-ippm-hybrid-two-step], have been proposed, which can provide flow information on the entire forwarding path on a per-packet basis in real-time. The aforementioned on-path telemetry techniques differ from the active and passive OAM schemes discussed earlier in that, they directly modify the user packets and can guarantee 100% accuracy. These on-path telemetry techniques can be classified as the OAM hybrid type I, since they involve "augmentation or modification of the stream of interest, or employment of methods that modify the treatment of the streams", according to [RFC7799].

On-path telemetry is useful for application-aware networking operations not only in data center and enterprise networks but also in carrier networks which may cross multiple domains. Carrier network operators have shown interest in utilizing such techniques for various purposes. For example, it is critical for the operators who offer high-bandwidth, latency and loss-sensitive services such as video streaming and online gaming to closely monitor the relevant flows in real-time as the basis for any further optimizations.

This framework document is intended to guide system designers attempting to use the referenced techniques as well as to motivate further work to enhance the telemetry ecosystem. It highlights requirements and challenges, outlines vital techniques that are applicable, and provides examples of how these might be applied for critical use cases.

The document scope is discussed in Section 1.3.

1.1. Classification and Modes of On-path Telemetry

The operation of on-path telemetry differs from both active OAM and passive OAM as defined in [RFC7799]. It does not generate any active probe packets or passively observes unmodified user packets. Instead, it modifies selected user packets to collect useful information about them. Therefore, the operation can be categorized as the hybrid OAM type I mode per [RFC7799], which can provide more flexible and accurate network monitoring and measurement.

This hybrid type OAM can be further partitioned into two modes. [passport-postcard] first uses the metaphor of "passport" and "postcard" to describe how the on-path data can be collected and exported. In the passport mode, each node on the path adds the telemetry data to the user packets (i.e., stamp the passport). The accumulated data trace is exported at a configured end node. In the postcard mode, each node directly exports the telemetry data using an independent packet (i.e., send a postcard) while the user packets are intact. It is possible to combine the two modes together in one solution. We call this the hybrid mode.

Figure 1 shows the classification of the existing on-path telemetry techniques.

Mode	Passport	Postcard	Hybrid
Technique	IOAM Trace IOAM E2E EAM	IOAM DEX PBT-M	Multicast Te- lemetry HTS

Figure 1: ON-path Telemetry Technique Classification

IOAM Trace and E2E options are described in [I-D.ietf-ippm-ioam-data]. EAM is described in [I-D.zhou-ippm-enhanced-alternate-marking]. IOAM DEX option is described in [I-D.ietf-ippm-ioam-direct-export] PBT-M is described in [I-D.song-ippm-postcard-based-telemetry]. Multicast Telemetry is described in [I-D.song-multicast-telemetry]. HTS is described in [I-D.mirsky-ippm-hybrid-two-step].

The advantages of the passport mode include :

o It naturally retains the telemetry data correlation along the entire path. The self-describing feature eases the data consumption.

o The on-path data for a packet is only exported once so the data export overhead is low.

o Only the head and end nodes of the paths need to be configured so the configuration overhead is low.

The disadvantages of the passport mode include :

o The telemetry data carried by user packets inflate the packet size, which is undesirable or prohibitive.

- o Approaches for encapsulating the instruction header and data in transport protocols need to be standardized, which is challenging.

- o Carrying sensitive data along the path is vulnerable to security and privacy breach.

- o If a packet is dropped on the path, the data collected so far are also lost.

The postcard mode provides a perfect complement to the passport mode by addressing the issues of the passport mode. The advantages of the postcard mode include:

- o Either there is no packet header overhead (e.g., PBT-M) or the overhead is small and fixed (e.g., IOAM DEX).

- o The encapsulation requirement can be avoided (e.g., PBT-M).

- o The telemetry data can be secured.

- o Even if a packet is dropped on the path, the partial data collected so far are still available.

The disadvantages of the postcard mode include:

- o Telemetry data are spread in multiple postcards so extra effort is needed to correlate the data.

- o Every node exports a postcard for a packet which increases the data export overhead.

- o In case of PBT-M, every node on the path needs to be configured, so the configuration overhead is high.

- o In case of IOAM DEX, the encapsulation requirement remains.

The hybrid mode either tailors for some specific application scenario (e.g., Multicast Telemetry) or provides some alternative approach (e.g., HTS).

1.2. Requirements and Challenges

Although on-path telemetry is beneficial, successfully applying such techniques in carrier networks must consider performance, deployability, and flexibility. Specifically, we need to address the following practical deployment challenges:

- C1: On-path telemetry incurs extra packet processing which may cause stress on the network data plane. The potential impact on the forwarding performance creates an unfavorable "observer effect". This will not only damages the fidelity of the measurement but also defies the purpose of the measurement. For example, the growing IOAM data per hop can negatively affect service levels by increasing the serialization delay and header parsing delay.

- C2: On-path telemetry can generate a considerable amount of data which may claim too much transport bandwidth and inundate the servers for data collection, storage, and analysis. Increasing the data handling capacity is technically viable but expensive. For example, if IOAM is applied to all the traffic, one node may collect a few tens of bytes as telemetry data for each packet. The whole forwarding path might accumulate a data-trace with a size similar to or even exceeding that of the original packet. Transporting the telemetry data alone is projected to consume almost half of the network bandwidth, plus it creates significant back-end data handling and storage requirements.

- C3: The collectible data defined currently are essential but limited. As the network operation evolves to be declarative (intent-based) and automated, and the trends of network virtualization, wireline and wireless convergence, and packet-optical integration continue, more data is needed in an on-demand and interactive fashion. Flexibility and extensibility on data defining, aggregation, acquisition, and filtering, must be considered.

- C4: Applying only a single underlying on-path telemetry technique may lead to a defective result. For example, packet drop can cause the loss of the flow telemetry data and the packet drop location and reason remains unknown if only the In-situ OAM trace option is used. A comprehensive solution needs the flexibility to switch between different underlying techniques and adjust the configurations and parameters at runtime. Thus, system-level orchestration is needed.

- C5: If we were to apply some on-path telemetry technique in today's carrier operator networks, we must provide solutions to tailor the provider's network deployment base and support an incremental deployment strategy. That is, we need to support established encapsulation schemes for various predominant protocols such as Ethernet, IPv4, IPv6, and MPLS with backward compatibility and properly handle various transport tunnels.

- C6: The development of simplified on-path telemetry primitives and models for configuration and queries is essential. Telemetry models may be utilized via an API-based telemetry service for external applications, for end-to-end performance measurement and application performance monitoring. The standard-based protocols and methods are needed for network configuration and programming, and telemetry data processing and export, to provide interoperability.

1.3. Scope

Following the network telemetry framework discussed in [I-D.ietf-opsawg-ntf], this document focuses on the on-path telemetry, a specific class of data-plane telemetry techniques, and provides a high-level framework which addresses the aforementioned challenges for deployment, especially in carrier operator networks.

This document aims to clarify the problem space, essential requirements, and summarizes best practices and general system design considerations. The framework helps to analyze the current standard status and identify gaps, and to motivate new standard works to complete the ecosystem. This document provides some examples to show some novel network telemetry applications under the framework.

As an informational document, it describes an open framework with a few key components. The framework does not enforces any specific implementation on each component, neither does it define interfaces (e.g., API, protocol) between components. The choice of underlying on-path telemetry techniques and other implementation details is determined by application implementer. Therefore, the framework is not a solution specification. It only provides a high-level overview and is not necessarily a mandatory recommendation for on-path telemetry applications. Implementation of the framework is implementor specific and may utilize functional components and techniques outlined in this document.

The standardization of the underlying techniques and interfaces mentioned in this document is undertaken by various working groups. Due to the limited scope and intended status of this document, it has no overlap or conflict with those works.

1.4. Glossary

This section defines and explains the acronyms and terms used in this document.

On-path Telemetry: Remotely acquiring performance and behavior data about network flows on a per-packet basis on the packet's forwarding path. The term refers to a class of data plane telemetry techniques, including IOAM, PBT, EAM, and HTS. Such techniques may need to mark user packets, or insert instruction or metadata to the headers of user packets.

IFIT: In-situ Flow Information Telemetry, pronounced as "I-Fit". The name of a high-level reference framework that shows how network data-plane monitoring applications can address the deployment challenges of the flow-oriented on-path telemetry techniques.

IFIT Domain: A network domain in which an on-path telemetry application operates. The network domain contains multiple forwarding devices, such as routers and switches, that are capable of IFIT-specific functions. It also contains a logically centralized controller whose responsibility is to apply IFIT-specific configurations and functions to IFIT-capable forwarding devices, and to collect and analyze the on-path telemetry data from those devices. An IFIT domain contains multiple network node capable of IFIT-specific functions. We name all the entry nodes to an IFIT domain head nodes and all the exit nodes end nodes. A path in an IFIT domain starts from a head node and ends at an end node. Usually the instruction header encapsulation or packet marking, if needed, happens at the head nodes; the instruction header decapsulation or packet unmarking, if needed, happens at the end nodes.

Reflective Telemetry: The telemetry functions in a dynamic and interactive fashion. A new telemetry action is provisioned as a result of self-knowledge acquired through prior telemetry actions.

1.5. 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. IFIT Overview

To address the challenges mentioned above, we present a high-level framework based on multiple network operators' requirements and common industry practice, which can help to build a workable and efficient on-path telemetry application. We name the framework "In-situ Flow Information Telemetry" (IFIT) to reflect the fact that this framework is dedicated to on-path telemetry data about user and application traffic flows. As a reference framework, IFIT covers a class of on-path telemetry techniques and works a level higher than any specific underlying technique. The framework is comprised of some key functional components (Section 3). By assembling these components, IFIT supports reflective telemetry that enables autonomous network operations (Section 4).

2.1. Typical Deployment of IFIT

Figure 2 shows a typical deployment scenario of IFIT.

```
                                   Application
                      +-------------------------------------+
                      |             Controller              |
                      | +------------+        +-----------+ |
                      | | Configure  |<-------| Collector | |
                      | |     &      |        |     &     | |
                      | | Control    |        | Analyzer  | |
                      | +-----:------+        +-----------+ |
                      |       :                     ^       |
                      +-------:---------------------|-------+
                              :configuration        |telemetry data
                              :& action             |
               ...............:.....................|..........
               :             :                 :    |         :
               :   +---------:---+-------------:---++---------:---+
               :   |         :   |             :   |          :   |
               V   |         V   |             V   |          V   |
            +------+-+     +-----+--+       +------+-+     +------+-+
     packets| Head   |     | Path   |       | Path   |     | End    |
         ==>| Node   |====>| Node   |==//==>| Node   |====>| Node   |==>
            |        |     | A      |       | B      |     |        |
            +--------+     +--------+       +--------+     +--------+

            |<---                  IFIT Domain                  --->|
```

Figure 2: IFIT Deployment Scenario

An on-path telemetry application can conduct some network data plane monitoring and measurement tasks over an IFIT domain by applying one or more underlying techniques. The application needs to contains multiple elements, including configuring the network nodes and processing the telemetry data. The application usually runs in a logically centralized controller which is responsible for configuring the network nodes in the IFIT domain, and collecting and analyzing telemetry data. The configuration determines which underlying technique is used, what telemetry data are of interest, which flows and packets are concerned with, how the telemetry data are collected, etc. The process can be dynamic and interactive: after the telemetry data processing and analyzing, the application may instruct the controller to modify the configuration of the nodes in the IFIT domain, which affects the future telemetry data collection.

From the system-level view, it is recommended to use the standardized configuration and data collection interfaces, regardless of the underlying technique. However, the specification of these interfaces and the implementation of the controller are out of scope for this document.

The IFIT domain encompasses the head nodes and the end nodes. An IFIT domain may cross multiple network domains. The head nodes are responsible for enabling the IFIT-specific functions and the end nodes are responsible for terminating them. All capable nodes in an IFIT domain will be capable of executing the instructed IFIT-specific function. It is important to note that any IFIT application must, through configuration and policy, guarantee that any packet with IFIT-specific header and metadata will not leak out of the IFIT domain. The end nodes must be able to capture all packets with IFIT-specific header and metadata and recover their format before forwarding them out of the IFIT domain.

The underlying on-path telemetry techniques covered by IFIT can be of any modes discussed in Section 1.1.

2.2. IFIT Architecture

The IFIT architecture is shown in Figure 3, which contains several key components. These components aim to address the deployment challenges discussed in Section 1. The detailed block diagram and description for each component are given in Section 3. Here we only provide a high level overview.

```
                  +-------------------------------------+
                  |        On-demand Technique          |
                  |        Selection & Integration      |
                  +-------------------------------------+
        Control Plane     |                   ^
   ---------------------+--------------------+-------------
        Forwarding Plane  V                   |
                  +------------------+------------------+
                  | Flexible Flow,   | Flexible         |
                  | Packet, & Data   | Data Export      |
                  | Selection        |                  |
                  +------------------+------------------|
                  |      Dynamic Network Probe          |
                  +-------------------------------------|
```

Figure 3: IFIT Architecture

Based on the monitoring and measurement requirements, an application needs to choose one or more underlying on-path telemetry techniques and decide the policies to apply them. Depending on the forwarding-plane protocol and tunneling configuration, the instruction header and metadata encapsulation method, if needed, is also determined. The encapsulation happens at the head nodes and the decapsulation happens at the end nodes.

Based on the network condition and application requirement, the head nodes also need to be able to choose flows and packets to enable the IFIT-specific functions, and decide the set of data to be collected. All the nodes who are responsible for exporting telemetry data are configured with special functions to prepare the data. The IFIT-specific functions can be deployed into the network nodes as dynamic network probes.

2.3. Relationship with Network Telemetry Framework (NTF)

[I-D.ietf-opsawg-ntf] describes a Network Telemetry Framework (NTF). One dimension used by NTF to partition network telemetry techniques and systems is based on the three planes in networks plus external data sources. IFIT fits in the category of forwarding-plane telemetry and deals with the specific on-path technical branch of the forwarding-plane telemetry.

According to NTF, an on-path telemetry application mainly subscribes event-triggered or streaming data. The key functional components of IFIT also match the components in NTF. "On-demand Technique Selection and Integration" is an application layer function, matching the "Data Query, Analysis, and Storage" component in NTF; "Flexible Flow, Packet, and Data Selection" matches the "Data Configuration and

Subscription" component; "Flexible Data Export" matches the "Data Encoding and Export" component; "Dynamic Network Probe" matches the "Data Generation and Processing" component.

3. Key Components of IFIT

As shown in the IFIT architecture, the key components of IFIT are as follows:

- Flexible flow, packet, and data selection policy, addressing the challenge C1 described in Section 1;

- Flexible data export, addressing the challenge C2;

- Dynamic network probe, addressing C3;

- On-demand technique selection and integration, addressing C4.

Note that the challenges C5 and C6 are mostly standard related, which are fundamental to IFIT. We discuss the standard status and gaps in Section 5.

In the following section, we provide a detailed description of each component.

3.1. Flexible Flow, Packet, and Data Selection

In most cases, it is impractical to enable the data collection for all the flows and for all the packets in a flow due to the potential performance and bandwidth impact. Therefore, a workable solution usually need to select only a subset of flows and flow packets to enable the data collection, even though this means the loss of some information and accuracy.

In the data plane, the Access Control List (ACL) provides an ideal means to determine the subset of flow(s). An application can set a sample rate or probability to a flow to allow only a subset of flow packets to be monitored, collect a different set of data for different packets, and disable or enable data collection on any specific network node. An application can further allow any node to accept or deny the data collection process in full or partially.

Based on these flexible mechanisms, IFIT allows applications to apply flexible flow and data selection policies to suit the requirements. The applications can dynamically change the policies at any time based on the network load, processing capability, focus of interest, and any other criteria.

3.1.1. Block Diagram

```
+-----------------------------+
| +-----------+  +-----------+ |
| |Flow       |  |Data       | |
| |Selection  |  |Selection  | |
| +-----------+  +-----------+ |
| +-----------+                |
| |Packet     |                |
| |Selection  |                |
| +-----------+                |
+-----------------------------+
```

Figure 4: Flexible Flow, Packet, and Data Selection

Figure 4 shows the block diagram of this component. The flow selection block defines the policies to choose target flows for monitoring. Flow has different granularity. A basic flow is defined by 5-tuple IP header fields. Flow can also be aggregated at interface level, tunnel level, protocol level, and so on. The packet selection block defines the policies to choose packets from a target flow. The policy can be either a sampling interval, a sampling probability, or some specific packet signature. The data selection block defines the set of data to be collected. This can be changed on a per-packet or per-flow basis.

3.1.2. Example: Sketch-guided Elephant Flow Selection

Network operators are usually more interested in elephant flows which consume more resource and are sensitive to changes in network conditions. A CountMin Sketch [CMSketch] can be used on the data path of the head nodes, which identifies and reports the elephant flows periodically. The controller maintains a current set of elephant flows and dynamically enables the on-path telemetry for only these flows.

3.1.3. Example: Adaptive Packet Sampling

Applying on-path telemetry on all packets of selected flows can still be out of reach. A sample rate should be set for these flows and only enable telemetry on the sampled packets. However, the head nodes have no clue on the proper sampling rate. An overly high rate would exhaust the network resource and even cause packet drops; An overly low rate, on the contrary, would result in the loss of information and inaccuracy of measurements.

An adaptive approach can be used based on the network conditions to dynamically adjust the sampling rate. Every node gives user traffic forwarding higher priority than telemetry data export. In case of network congestion, the telemetry can sense some signals from the data collected (e.g., deep buffer size, long delay, packet drop, and data loss). The controller may use these signals to adjust the packet sampling rate. In each adjustment period (i.e., RTT of the feedback loop), the sampling rate is either decreased or increased in response of the signals. An AIMD policy similar to the TCP flow control mechanism for the rate adjustment can be used.

3.2. Flexible Data Export

The flow telemetry data can catch the dynamics of the network and the interactions between user traffic and network. Nevertheless, the data inevitably contain redundancy. It is advisable to remove the redundancy from the data in order to reduce the data transport bandwidth and server processing load.

In addition to efficient export data encoding (e.g., IPFIX [RFC7011] or protobuf [1]), nodes have several other ways to reduce the export data by taking advantage of network device’s capability and programmability. Nodes can cache the data and send the accumulated data in batch if the data is not time sensitive. Various deduplication and compression techniques can be applied on the batch data.

From the application perspective, an application may only be interested in some special events which can be derived from the telemetry data. For example, in case that the forwarding delay of a packet exceeds a threshold, or a flow changes its forwarding path is of interest, it is unnecessary to send the original raw data to the data collecting and processing servers. Rather, IFIT takes advantage of the in-network computing capability of network devices to process the raw data and only push the event notifications to the subscribing applications.

Such events can be expressed as policies. An policy can request data export only on change, on exception, on timeout, or on threshold.

3.2.1. Block Diagram

```
                 +-------------------------------------------+
                 | +-----------+ +-----------+ +-----------+ |
                 | |Data       | |Data       | |Export     | |
                 | |Encoding   | |Batching   | |Protocol   | |
                 | +-----------+ +-----------+ +-----------+ |
                 | +-----------+ +-----------+ +-----------+ |
                 | |Data       | |Data       | |Data       | |
                 | |Compression| |Dedup.     | |Filter     | |
                 | +-----------+ +-----------+ +-----------+ |
                 | +-----------+ +-----------+               |
                 | |Data       | |Data       |               |
                 | |Computing  | |Aggregation|               |
                 | +-----------+ +-----------+               |
                 +-------------------------------------------+
```

Figure 5: Flexible Data Export

Figure 5 shows the block diagram of this component. The data encoding block defines the method to encode the telemetry data. The data batching block defines the size of batch data buffered at the device side before export. The export protocol block defines the protocol used for telemetry data export. The data compression block defines the algorithm to compress the raw data. The data deduplication block defines the algorithm to remove the redundancy in the raw data. The data filter block defines the policies to filter the needed data. The data computing block defines the policies to prepocess the raw data and generate some new data. The data aggregation block defines the procedure to combine and synthesize the data.

3.2.2. Example: Event-based Anomaly Monitor

Network operators are interested in the anomalies such as path change, network congestion, and packet drop. Such anomalies are hidden in raw telemetry data (e.g., path trace, timestamp). Such anomalies can be described as events and programmed into the device data plane. Only the triggered events are exported. For example, if a new flow appears at any node, a path change event is triggered; if the packet delay exceeds a predefined threshold in a node, the congestion event is triggered; if a packet is dropped due to buffer overflow, a packet drop event is triggered.

The export data reduction due to such optimization is substantial. For example, given a single 5-hop 10Gbps path, assume a moderate number of 1 million packets per second are monitored, and the telemetry data plus the export packet overhead consume less than 30 bytes per hop. Without such optimization, the bandwidth consumed by the telemetry data can easily exceed 1Gbps (>10% of the path bandwidth), When the optimization is used, the bandwidth consumed by the telemetry data is negligible. Moreover, the pre-processed telemetry data greatly simplify the work of data analyzers.

3.3. Dynamic Network Probe

Due to limited data plane resource and network bandwidth, it is unlikely one can monitor all the data all the time. On the other hand, the data needed by applications may be arbitrary but ephemeral. It is critical to meet the dynamic data requirements with limited resource.

Fortunately, data plane programmability allows IFIT to dynamically load new data probes. These on-demand probes are called Dynamic Network Probes (DNP). DNP is the technique to enable probes for customized data collection in different network planes. When working with IOAM or PBT, DNP is loaded to the data plane through incremental programming or configuration. The DNP can effectively conduct data generation, processing, and aggregation.

DNP introduces enough flexibility and extensibility to IFIT. It can implement the optimizations for export data reduction motioned in the previous section. It can also generate custom data as required by today and tomorrow's applications.

3.3.1. Block Diagram

```
+----------------------------+
| +-----------+  +-----------+ |
| |ACL        |  |YANG       | |
| |           |  |Model      | |
| +-----------+  +-----------+ |
| +-----------+  +-----------+ |
| |Hardware   |  |Software   | |
| |Function   |  |Function   | |
| +-----------+  +-----------+ |
+----------------------------+
```

Figure 6: Dynamic Network Probes

Figure 6 shows the block diagram of this component. The Access Control List (ACL) block is available in most hardware and it defines DNPs through dynamically update the ACL policies (including flow filtering and action). YANG models can be dynamically deployed to enable different data processing and filtering functions. Some hardware allows dynamically loading hardware-based functions into the forwarding path at runtime through mechanisms such as reserved pipelines and function stubs. Dynamically loadable software functions can be implemented in the control processors in IFIT nodes.

3.3.2. Examples

Following are some possible DNPs that can be dynamically deployed to support applications.

On-demand Flow Sketch: A flow sketch is a compact online data structure (usually a variation of multi-hashing table) for approximate estimation of multiple flow properties. It can be used to facilitate flow selection. The aforementioned CountMin Sketch [CMSketch] is such an example. Since a sketch consumes data plane resources, it should only be deployed when actually needed.

Smart Flow Filter: The policies that choose flows and packet sampling rate can change during the lifetime of an application.

Smart Statistics: An application may need to count flows based on different flow granularity or maintain hit counters for selected flow table entries.

Smart Data Reduction: DNP can be used to program the events that conditionally trigger data export.

3.4. On-demand Technique Selection and Integration

With multiple underlying data collection and export techniques at its disposal, IFIT can flexibly adapt to different network conditions and different application requirements.

For example, depending on the types of data that are of interest, IFIT may choose either IOAM or PBT to collect the data; if an application needs to track down where the packets are lost, switching from IOAM to PBT should be supported.

IFIT can further integrate multiple data plane monitoring and measurement techniques together and present a comprehensive data plane telemetry solution.

Based on the application requirements and the real-time telemetry data analysis results, new configurations and actions can be deployed.

3.4.1. Block Diagram

```
     +-----------------------------------------------+
     | +------------+  +-------------+  +---------+ |
     | |Application |  |Configuration|  |Telemetry| |
     | |Requirements|->|& Action     |<-|Data     | |
     | |            |  |             |  |Analysis | |
     | +------------+  +-------------+  +---------+ |
     +-----------------------------------------------+
     | Passport Mode:                                |
     | +----------+   +----------+   +----------+    |
     | |IOAM E2E  |   |IOAM Trace|   |EAM       |    |
     | +----------+   +----------+   +----------+    |
     | Postcard Mode:                                |
     | +----------+   +----------+                   |
     | |PBT-M     |   |IOAM DEX  |                   |
     | +----------+   +----------+                   |
     | Hybrid Mode:                                  |
     | +----------+   +----------+                   |
     | |HTS       |   |Multicast |                   |
     | |          |   |Telemetry |                   |
     | +----------+   +----------+                   |
     +-----------------------------------------------+
```

Figure 7: Technique Selection and Integration

Figure 7 shows the block diagram of this component, which lists the candidate on-path telemetry techniques.

Located in the logically centralized controller of an IFIT domain, this component makes all the control and configuration dynamically to the capable nodes in the domain which will affect the future telemetry data. The configuration and action decisions are based on the inputs from the application requirements and the realtime telemetry data analysis results. Note that here the telemetry data source is not limited to the data plane. The data can come form all the sources mentioned in [I-D.ietf-opsawg-ntf], including external data sources.

4. IFIT for Reflective Telemetry

The IFIT components can work together to support reflective telemetry, as shown in Figure 8.

```
             +---------------------+
             |                     |
      +------+    Applications     |<------+
      |      |                     |       |
      |      +---------------------+       |
      |         Technique Selection        |
      |         and Integration            |
      |                                    |
      |Flexible                   Flexible |
      |Flow,     reflection-loop      Data |
      |Packet,                      Export |
      |and Data                            |
      |Selection                     +----+----+
      V                             +---------+|
+----------+ Encapsulation         +---------+||
|  Head    | and Tunneling         |  Path   |||
|  Node    |---------------------->|  Nodes  ||+
|          |                       |         |+
+----------+                       +---------+
    DNP                               DNP
```

Figure 8: IFIT-based Reflective Telemetry

An application may pick a suite of telemetry techniques based on its requirements and apply an initial technique to the data plane. It then configures the head nodes to decide the initial target flows/ packets and telemetry data set, the encapsulation and tunneling scheme based on the underlying network architecture, and the IFIT-capable nodes to decide the initial telemetry data export policy. Based on the network condition and the analysis results of the telemetry data, the application can change the telemetry technique, the flow/data selection policy, and the data export approach in real time without breaking the normal network operation. Many of such dynamic changes can be done through loading and unloading DNPs.

The reflective telemetry enabled by the IFIT allows numerous new applications suitable for future network operation architecture.

4.1. Example: Intelligent Multipoint Performance Monitoring

[I-D.ietf-ippm-multipoint-alt-mark] describes an intelligent performance management based on the network condition. The idea is to split the monitoring network into clusters. The cluster partition that can be applied to every type of network graph and the possibility to combine clusters at different levels enable the so-called Network Zooming. It allows a controller to calibrate the network telemetry, so that it can start without examining in depth and monitor the network as a whole. In case of necessity (packet loss or too high delay), an immediate detailed analysis can be reconfigured. In particular, the controller, that is aware of the network topology, can set up the most suited cluster partition by changing the traffic filter or activate new measurement points and the problem can be localized with a step-by-step process.

An application on top of the controllers can manage such mechanism and IFIT's architecture allows its dynamic and reflective operation.

4.2. Example: Intent-based Network Monitoring

```
                         User Intents
                               |
                               V          Per-packet
                         +------------+   Telemetry
                  ACL    |            |   Data
                +--------+ Controller |<--------+
                |        |            |         |
                |        +--+---------+         |
                |           |       ^           |
                |           |DNPs   |Network    |
                |           |       |Information|
                |           V       |           |
         +------+-------------------+-----------+---+
         |      |                                   |
         |      V                      +------+     |
         | +-------+                  +------+|     |
         | | Head  |    IFIT Domain  +------+||     |
         | | Node  |                 |Path  ||+     |
         | |       |                 |Nodes |+      |
         | +-------+                 +------+       |
         +------------------------------------------+
```

Figure 9: Intent-based Monitoring

In this example, a user can express high level intents for network monitoring. The controller translates an intent and configure the corresponding DNPs in IFIT-capable nodes which collect necessary network information. Based on the real-time information feedback, the controller runs a local algorithm to determine the suspicious flows. It then deploys ACLs to the head node to initiate the high precision per-packet on-path telemetry for these flows.

5. Standard Status and Gaps

A complete IFIT-based solution needs standard interfaces for configuration and data extraction, and standard encapsulation on various transport protocols. It may also need standard API and primitives for application programming and deployment. The draft [I-D.brockners-opsawg-ioam-deployment] summarizes some current proposals on encapsulation and data export for IOAM. These works should be extended or modified to support other types of on-path telemetry techniques and other transport protocols. The high-level IFIT helps to develop coherent and universal standard encapsulation and data export approaches.

5.1. Encapsulation in Transport Protocols

Since the introduction of IOAM, the IOAM option header encapsulation schemes in various network protocols have been proposed. Similar encapsulation schemes need to be extended to cover the other on-path telemetry techniques. On the other hand, the encapsulation scheme for some popular protocols, such as MPLS and IPv4, are noticeably missing. It is important to provide the encapsulation schemes for these protocols because they are still prevalent in carrier networks. IFIT needs to provide solutions to apply the on-path flow telemetry techniques in such networks. PBT-M [I-D.song-ippm-postcard-based-telemetry] does not introduce new headers to the packets so the trouble of encapsulation for a new header is avoided. While there are some proposals which allow new header encapsulation in MPLS packets (e.g., [I-D.song-mpls-extension-header]) or in IPv4 packets (e.g., [I-D.herbert-ipv4-eh]), they are still in their infancy stage and require significant future work. For the meantime, in a confined IFIT domain, pre-standard encapsulation approaches may be applied.

5.2. Tunneling Support

In carrier networks, it is common for user traffic to traverse various tunnels for QoS, traffic engineering, or security. IFIT supports both the uniform mode and the pipe mode for tunnel support as described in [I-D.song-ippm-ioam-tunnel-mode]. With such flexibility, the operator can either gain a true end-to-end visibility or apply a hierarchical approach which isolates the monitoring domain between customer and provider.

5.3. Deployment Automation

In addition, standard approaches that automates the function configuration, and capability query and advertisement, either in a centralized fashion or a distributed fashion, are still immature.

The draft [I-D.zhou-ippm-ioam-yang] provides the YANG model for IOAM configuration. Similar models needs to be defined for other techniques. It is also helpful to provide standards-based approaches for configuration in various network environments. For example, in segment routing networks, extensions to BGP or PCEP can be defined to distribute SR policies carrying IFIT information, so that IFIT behavior can be enabled automatically when the SR policy is applied. [I-D.chen-pce-sr-policy-ifit] proposes to extend PCEP policy for IFIT configuration in segment routing networks. [I-D.qin-idr-sr-policy-ifit] proposes to extend BGP policy instead for IFIT configuration in segment routing networks. Additional capability discovery and dissemination will be needed for other types of networks.

To realize the potential of IFIT, programming and deploying DNPs are important. ForCES [RFC5810] is a standard protocol for network device programming, which can be used for DNP deployment. Currently some related works such as [I-D.wwx-netmod-event-yang] and [I-D.bwd-netmod-eca-framework] have proposed to use YANG model to define the smart policies which can be used to implement DNPs. In the future, other approaches for hardware and software-based functions can be development to enhance the programmability and flexibility.

6. Summary

IFIT is a high-level framework for applying on-path telemetry techniques, and this document has outlined how the framework may be used to solve essential use cases. IFIT enables a practical data-plane telemetry application based on two basic on-path traffic data collection modes: passport and postcard.

IFIT addresses the key challenges for operators to deploy a complete on-path telemetry solution. However, as a reference and open framework, IFIT only describes the basic functions of each identified component and suggests possible applications. It has no intention of specifying the implementation of the components and the interfaces between the components. The compliance of IFIT is by no means mandatory either. Instead, this informational document aims to clarify the problem domain, and summarize the best practices and sensible system design considerations. IFIT can guide the analysis of the current standard status and gaps, and motivate new works to complete the ecosystem. IFIT enables data-plane reflective telemetry applications for advanced network operations.

Having a high-level framework covering a class of related techniques also promotes a holistic approach for standard development and helps to avoid duplicated efforts and piecemeal solutions that only focus on a specific technique while omitting the compatibility and extensibility issues, which is important to a healthy ecosystem for network telemetry.

7. Security Considerations

In addition to the specific security issues discussed in each individual document on on-path telemetry, this document considers the overall security issues at the IFIT system level. This should serve as a guide to the on-path telemetry application developers and users.

8. IANA Considerations

This document includes no request to IANA.

9. Contributors

Other major contributors of this document include Giuseppe Fioccola, Daniel King, Zhenqiang Li, Zhenbin Li, Tianran Zhou, and James Guichard.

10. Acknowledgments

We thank Diego Lopez, Shwetha Bhandari, Joe Clarke, Adrian Farrel, Frank Brockners, Al Morton, Alex Clemm, Alan DeKok, and Warren Kumari for their constructive suggestions for improving this document.

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>.

[RFC7799] Morton, A., "Active and Passive Metrics and Methods (with Hybrid Types In-Between)", RFC 7799, DOI 10.17487/RFC7799, May 2016, <https://www.rfc-editor.org/info/rfc7799>.

[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>.

11.2. Informative References

[CMSketch]
Cormode, G. and S. Muthukrishnan, "An improved data stream summary: the count-min sketch and its applications", 2005, <http://dx.doi.org/10.1016/j.jalgor.2003.12.001>.

[I-D.brockners-opsawg-ioam-deployment]
Brockners, F., Bhandari, S., and d. daniel.bernier@bell.ca, "In-situ OAM Deployment", draft-brockners-opsawg-ioam-deployment-02 (work in progress), September 2020.

[I-D.bwd-netmod-eca-framework]
Boucadair, M., WU, Q., Wang, Z., King, D., and C. Xie, "Framework for Use of ECA (Event Condition Action) in Network Self Management", draft-bwd-netmod-eca-framework-00 (work in progress), November 2019.

[I-D.chen-pce-sr-policy-ifit]
Chen, H., Yuan, H., Zhou, T., Li, W., Fioccola, G., and Y. Wang, "PCEP SR Policy Extensions to Enable IFIT", draft-chen-pce-sr-policy-ifit-02 (work in progress), July 2020.

[I-D.herbert-ipv4-eh]
Herbert, T., "IPv4 Extension Headers and Flow Label", draft-herbert-ipv4-eh-01 (work in progress), May 2019.

[I-D.ietf-ippm-ioam-data]
Brockners, F., Bhandari, S., and T. Mizrahi, "Data Fields for In-situ OAM", draft-ietf-ippm-ioam-data-10 (work in progress), July 2020.

[I-D.ietf-ippm-ioam-direct-export]
Song, H., Gafni, B., Zhou, T., Li, Z., Brockners, F., Bhandari, S., Sivakolundu, R., and T. Mizrahi, "In-situ OAM Direct Exporting", draft-ietf-ippm-ioam-direct-export-01 (work in progress), August 2020.

[I-D.ietf-ippm-multipoint-alt-mark]
Fioccola, G., Cociglio, M., Sapio, A., and R. Sisto, "Multipoint Alternate Marking method for passive and hybrid performance monitoring", draft-ietf-ippm-multipoint-alt-mark-09 (work in progress), March 2020.

[I-D.ietf-opsawg-ntf]
Song, H., Qin, F., Martinez-Julia, P., Ciavaglia, L., and A. Wang, "Network Telemetry Framework", draft-ietf-opsawg-ntf-04 (work in progress), September 2020.

[I-D.mirsky-ippm-hybrid-two-step]
Mirsky, G., Lingqiang, W., and G. Zhui, "Hybrid Two-Step Performance Measurement Method", draft-mirsky-ippm-hybrid-two-step-05 (work in progress), April 2020.

[I-D.qin-idr-sr-policy-ifit]
Qin, F., Yuan, H., Zhou, T., Fioccola, G., and Y. Wang, "BGP SR Policy Extensions to Enable IFIT", draft-qin-idr-sr-policy-ifit-04 (work in progress), October 2020.

[I-D.song-ippm-ioam-tunnel-mode]
Song, H., Li, Z., Zhou, T., and Z. Wang, "In-situ OAM Processing in Tunnels", draft-song-ippm-ioam-tunnel-mode-00 (work in progress), June 2018.

[I-D.song-ippm-postcard-based-telemetry]
Song, H., Zhou, T., Li, Z., Shin, J., and K. Lee, "Postcard-based On-Path Flow Data Telemetry", draft-song-ippm-postcard-based-telemetry-07 (work in progress), April 2020.

[I-D.song-mpls-extension-header]
Song, H., Li, Z., Zhou, T., and L. Andersson, "MPLS Extension Header", draft-song-mpls-extension-header-02 (work in progress), February 2019.

[I-D.song-multicast-telemetry]
Song, H., McBride, M., and G. Mirsky, "Requirement and Solution for Multicast Traffic On-path Telemetry", draft-song-multicast-telemetry-04 (work in progress), April 2020.

[I-D.wwx-netmod-event-yang]
Bierman, A., WU, Q., Bryskin, I., Birkholz, H., Liu, X., and B. Claise, "A YANG Data model for ECA Policy Management", draft-wwx-netmod-event-yang-09 (work in progress), July 2020.

[I-D.zhou-ippm-enhanced-alternate-marking]
Zhou, T., Fioccola, G., Lee, S., Cociglio, M., and W. Li, "Enhanced Alternate Marking Method", draft-zhou-ippm-enhanced-alternate-marking-05 (work in progress), July 2020.

[I-D.zhou-ippm-ioam-yang]
Zhou, T., Guichard, J., Brockners, F., and S. Raghavan, "A YANG Data Model for In-Situ OAM", draft-zhou-ippm-ioam-yang-08 (work in progress), July 2020.

[passport-postcard]
Handigol, N., Heller, B., Jeyakumar, V., Mazieres, D., and N. McKeown, "Where is the debugger for my software-defined network?", 2012, <https://doi.org/10.1145/2342441.2342453>.

[RFC5810] Doria, A., Ed., Hadi Salim, J., Ed., Haas, R., Ed., Khosravi, H., Ed., Wang, W., Ed., Dong, L., Gopal, R., and J. Halpern, "Forwarding and Control Element Separation (ForCES) Protocol Specification", RFC 5810, DOI 10.17487/RFC5810, March 2010, <https://www.rfc-editor.org/info/rfc5810>.

[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>.

11.3. URIs

[1] https://developers.google.com/protocol-buffers/

Authors' Addresses

Haoyu Song
Futurewei
2330 Central Expressway
Santa Clara
USA

Email: haoyu.song@futurewei.com

Fengwei Qin
China Mobile
No. 32 Xuanwumenxi Ave., Xicheng District
Beijing, 100032
P.R. China

Email: qinfengwei@chinamobile.com

Huanan Chen
China Telecom
P. R. China

Email: chenhuan6@chinatelecom.cn

Jaehwan Jin
LG U+
South Korea

Email: daenamu1@lguplus.co.kr

Jongyoon Shin
SK Telecom
South Korea

Email: jongyoon.shin@sk.com

OPSAWG
Internet-Draft
Intended status: Informational
Expires: September 7, 2019

H. Song, Ed.
Huawei
ZQ. Li
China Mobile
P. Martinez-Julia
NICT
L. Ciavaglia
Nokia
A. Wang
China Telecom
March 6, 2019

Network Telemetry Framework
draft-song-opsawg-ntf-03

Abstract

This document provides an architectural framework for network telemetry to address the current and future network operation challenges and requirements. As evidenced by the defining characteristics and industry practice, network telemetry covers technologies and protocols beyond the conventional network Operations, Administration, and Management (OAM). Network telemetry promises better flexibility, scalability, accuracy, coverage, and performance and allows automated control loops to suit both today's and tomorrow's network operation requirements. This document clarifies the terminologies and classifies the modules and components of a network telemetry system. The framework and taxonomy help to set a common ground for the collection of related work and provide guidance for future technique and standard developments.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Network visibility is essential for network operation. Network telemetry has been widely considered as an ideal mean to gain sufficient network visibility with better flexibility, scalability, accuracy, coverage, and performance than conventional OAM technologies. However, confusion and misunderstandings about the network telemetry remain (e.g., the scope and coverage of the term). We need an unambiguous concept and a clear architectural framework for network telemetry so we can better align the related technology and standard work.

First, we show some key characteristics of network telemetry which set a clear distinction from the conventional network OAM and show that some conventional OAM technologies can be considered a subset of the network telemetry technologies. We then provide an architectural framework for network telemetry to meet the current and future network operation requirements. Following the framework, we classify the components of a network telemetry system so we can easily map the existing and emerging techniques and protocols into the framework. At last, we outline a roadmap for the evolution of the network telemetry system.

The purpose of the framework and taxonomy is to set a common ground for the collection of related work and provide guidance for future technique and standard developments.

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. Motivation

Thanks to the advance of the computing and storage technologies, today's big data analytics and machine learning-based Artificial Intelligence (AI) give network operators an unprecedented opportunity to gain network insights and move towards network autonomy. Software tools can use the network data to detect and react on network faults, anomalies, and policy violations, as well as predicting future events. In turn, the network policy updates for planning, intrusion prevention, optimization, and self-healing may be applied.

It is conceivable that an intent-driven autonomous network is the logical next step for network evolution following Software Defined Network (SDN), aiming to reduce (or even eliminate) human labor, make the most efficient use of network resources, and provide better services more aligned with customer requirements. Although it takes time to reach the ultimate goal, the journey has started nevertheless.

However, the system bottleneck is shifting from data consumption to data supply. Both the number of network nodes and the traffic bandwidth keep increasing at a fast pace; The network configuration and policy change at a much smaller time frame than ever before; More subtle events and fine-grained data through all network planes need to be captured and exported in real time. In a nutshell, it is challenging to get enough high-quality data out of network efficiently, timely, and flexibly. Therefore, we need to examine the existing network technologies and protocols, and identify any potential gaps based on the real network and device architectures.

In the remaining of this section, first we discuss several key use cases for today's and future network operations. Next, we show why the current network OAM techniques and protocols are insufficient for these use cases. The discussion underlines the need for new methods, techniques, and protocols which we may assign under an umbrella term - network telemetry.

2.1. Use Cases

These use cases are essential for network operations. While the list is by no means exhaustive, it is enough to highlight the requirements for data velocity, variety, and volume in networks.

Policy and Intent Compliance: Network policies are the rules that constraint the services for network access, provide service differentiation, or enforce specific treatment on the traffic. For example, a service function chain is a policy that requires the selected flows to pass through a set of ordered network functions. An intents is a high-level abstract policy which requires a complex translation and mapping process before being applied on networks. While a policy is enforced, the compliance needs to be verified and monitored continuously.

SLA Compliance: A Service-Level Agreement (SLA) defines the level of service a user expects from a network operator, which include the metrics for the service measurement and remedy/penalty procedures when the service level misses the agreement. Users need to check if they get the service as promised and network operators need to evaluate how they can deliver the services that can meet the SLA.

Root Cause Analysis: Any network failure can be the cause or effect of a sequence of chained events. Troubleshooting and recovery require quick identification of the root cause of any observable issues. However, the root cause is not always straightforward to identify, especially when the failure is sporadic and the related and unrelated events are overwhelming. While machine learning technologies can be used for root cause analysis, it up to the network to sense and provide all the relevant data.

Network Optimization: This covers all short-term and long-term network optimization techniques, including load balancing, Traffic Engineering (TE), and network planning. Network operators are motivated to optimize their network utilization and differentiate services for better ROI or lower CAPEX. The first step is to know the real-time network conditions before applying policies for traffic manipulation. In some cases, micro-bursts need to be detected in a very short time-frame so that fine-grained traffic control can be applied to avoid network congestion. The long-term network capacity planning and topology augmentation also rely on the accumulated data of the network operations.

Event Tracking and Prediction: The visibility of user traffic path and performance is critical for healthy network operation. Numerous related network events are of interest to network operators. For example, Network operators always want to learn where and why packets are dropped for an application flow. They also want to be warned of issues in advance so proactive actions can be taken to avoid catastrophic consequences.

2.2. Challenges

For a long time, network operators have relied upon SNMP [RFC3416], Command-Line Interface (CLI), or Syslog to monitor the network. Some other OAM techniques as described in [RFC7276] are also used to facilitate network troubleshooting. These conventional techniques are not sufficient to support the above use cases for the following reasons:

- Most use cases need to continuously monitor the network and dynamically refine the data collection in real-time and interactively. The poll-based low-frequency data collection is ill-suited for these applications. Subscription-based streaming data directly pushed from the data source (e.g., the forwarding chip) is preferred to provide enough data quantity and precision at scale.

- Comprehensive data is needed from packet processing engine to traffic manager, from line cards to main control board, from user flows to control protocol packets, from device configurations to operations, and from physical layer to application layer. Conventional OAM only covers a narrow range of data (e.g., SNMP only handles data from the Management Information Base (MIB)). Traditional network devices cannot provide all the necessary probes. An open and programmable network device is therefore needed.

- Many application scenarios need to correlate data from multiple sources (i.e., from distributed network devices, different components of a network device, or different network planes). A piecemeal solution is often lacking the capability to consolidate the data from multiple sources. The composition of a complete solution, as partly proposed by Autonomic Resource Control Architecture(ARCA) [I-D.pedro-nmrg-anticipated-adaptation], will be empowered and guided by a comprehensive framework.

- Some of the conventional OAM techniques (e.g., CLI and Syslog) are lack of formal data model. The unstructured data hinder the tool automation and application extensibility. Standardized data models are essential to support the programmable networks.

- Although some conventional OAM techniques support data push (e.g., SNMP Trap [RFC2981][RFC3877], Syslog, and sFlow), the pushed data are limited to only predefined management plane warnings (e.g., SNMP Trap) or sampled user packets (e.g., sFlow). We require the data with arbitrary source, granularity, and precision which are beyond the capability of the existing techniques.

- The conventional passive measurement techniques can either consume too much network resources and render too much redundant data, or lead to inaccurate results; the conventional active measurement techniques can interfere with the user traffic and their results are indirect. We need techniques that can collect direct and on-demand data from user traffic.

2.3. Glossary

Before further discussion, we list some key terminology and acronyms used in this documents. We make an intended distinction between network telemetry and network OAM.

AI: Artificial Intelligence. Use machine-learning based technologies to automate network operation.

BMP: BGP Monitoring Protocol

DNP: Dynamic Network Probe

DPI: Deep Packet Inspection

gNMI: gRPC Network Management Interface

gRPC: gRPC Remote Procedure Call

IDN: Intent-Driven Network

IPFIX: IP Flow Information Export Protocol

IPFPM: IP Flow Performance Measurement

IOAM: In-situ OAM

NETCONF: Network Configuration Protocol

Network Telemetry: Acquiring network data remotely for network monitoring and operation. A general term for a large set of network visibility techniques and protocols, with the characteristics defined in this document. Network telemetry addresses the current network operation issues and enables smooth evolution toward intent-driven autonomous networks.

NMS: Network Management System

OAM: Operations, Administration, and Maintenance. A group of network management functions that provide network fault indication, fault localization, performance information, and data and diagnosis functions. Most conventional network monitoring techniques and protocols belong to network OAM.

SNMP: Simple Network Management Protocol

YANG: A data modeling language for NETCONF

YANG FSM: A YANG model to define device side finite state machine

YANG PUSH: A method to subscribe pushed data from remote YANG datastore

2.4. Network Telemetry

Network telemetry has emerged as a mainstream technical term to refer to the newer data collection and consumption techniques, distinguishing itself form the convention techniques for network OAM. The representative techniques and protocols include IPFIX [RFC7011] and gPRC [I-D.kumar-rtgwg-grpc-protocol]. Network telemetry allows separate entities to acquire data from network devices so that data can be visualized and analyzed to support network monitoring and operation. Network telemetry overlaps with the conventional network OAM and has a wider scope than it. It is expected that network telemetry can provide the necessary network insight for autonomous networks, address the shortcomings of conventional OAM techniques, and allow for the emergence of new techniques bearing certain characteristics.

One difference between the network telemetry and the network OAM is that the network telemetry assumes machines as data consumer, while the conventional network OAM usually assumes human operators. Hence, the network telemetry can directly trigger the automated network operation, but the conventional OAM tools only help human operators to monitor and diagnose the networks and guide manual network operations. The difference leads to very different techniques.

Although the network telemetry techniques are just emerging and subject to continuous evolution, several characteristics of network telemetry have been well accepted (Note that network telemetry is intended to be an umbrella term covering a wide spectrum of techniques, so the following characteristics are not expected to be held by every specific technique):

o Push and Streaming: Instead of polling data from network devices, the telemetry collector subscribes to the streaming data pushed from data sources in network devices.

o Volume and Velocity: The telemetry data is intended to be consumed by machine rather than by a human. Therefore, the data volume is huge and the processing is often in realtime.

o Normalization and Unification: Telemetry aims to address the overall network automation needs. The piecemeal solutions offered by the conventional OAM approach are no longer suitable. Efforts need to be made to normalize the data representation and unify the protocols.

- Model-based: The telemetry data is modeled in advance which allows applications to configure and consume data with ease.

- Data Fusion: The data for a single application can come from multiple data sources (e.g., cross-domain, cross-device, and cross-layer) and needs to be correlated to take effect.

- Dynamic and Interactive: Since the network telemetry means to be used in a closed control loop for network automation, it needs to run continuously and adapt to the dynamic and interactive queries from the network operation controller.

Note that a technique does not need to have all the above characteristics to be qualified as telemetry. An ideal network telemetry solution may also have the following features or properties:

- In-Network Customization: The data can be customized in network at run-time to cater to the specific need of applications. This needs the support of a programmable data plane which allows probes to be deployed at flexible locations.

- Direct Data Plane Export: The data originated from data plane can be directly exported to the data consumer for efficiency, especially when the data bandwidth is large and the real-time processing is required.

- In-band Data Collection: In addition to the passive and active data collection approaches, the new hybrid approach allows to directly collect data for any target flow on its entire forwarding path.

- Non-intrusive: The telemetry system should avoid the pitfall of the "observer effect". That is, it should not change the network behavior and affect the forwarding performance.

3. The Necessity of a Network Telemetry Framework

Big data analytics and machine-learning based AI technologies are applied for network operation automation, relying on abundant data from networks. The single-sourced and static data acquisition cannot meet the data requirements. It is desirable to have a framework that integrates multiple telemetry approaches from different layers. This allows flexible combinations for different applications. The framework would benefit application development for the following reasons:

- o The future autonomous networks will require a holistic view on network visibility. All the use cases and applications need to be supported uniformly and coherently under a single intelligent agent. Therefore, the protocols and mechanisms should be consolidated into a minimum yet comprehensive set. A telemetry framework can help to normalize the technique developments.

- o Network visibility presents multiple viewpoints. For example, the device viewpoint takes the network infrastructure as the monitoring object from which the network topology and device status can be acquired; the traffic viewpoint takes the flows or packets as the monitoring object from which the traffic quality and path can be acquired. An application may need to switch its viewpoint during operation. It may also need to correlate a service and impact on network experience to acquire the comprehensive information.

- o Applications require network telemetry to be elastic in order to efficiently use the network resource and reduce the performance impact. Routine network monitoring covers the entire network with low data sampling rate. When issues arise or trends emerge, the telemetry data source can be modified and the data rate can be boosted.

- o Efficient data fusion is critical for applications to reduce the overall quantity of data and improve the accuracy of analysis.

So far, some telemetry related work has been done within IETF. However, the work is fragmented and scattered in different working groups. The lack of coherence makes it difficult to assemble a comprehensive network telemetry system and causes repetitive and redundant work.

A formal network telemetry framework is needed for constructing a working system. The framework should cover the concepts and components from the standardization perspective. This document clarifies the layers on which the telemetry is exerted and decomposes the telemetry system into a set of distinct components that the existing and future work can easily map to.

4. Network Telemetry Framework

Network telemetry techniques can be classified from multiple dimensions. In this document, we provide three unique perspectives: data acquiring mechanisms, data objects, and function components.

4.1. Data Acquiring Mechanisms

Broadly speaking, network data can be acquired through subscription (push) and query (poll). A subscriber may request data when it is ready. It follows a pub-sub mode or a sub-pub mode. In the pub-sub mode, pre-defined data are published and multiple qualified subscribers can subscribe the data. In the sub-pub mode, a subscriber designates what data are of interest and demands the network devices to deliver the data when they are avaiable.

In contrast, a querier expects immediate feedback from network devices. It is usually used in a more interactive environment. The queried data may be directly extracted from some specific data source, or synthesized and processed from raw data.

There are four types of data from network devices:

Simple Data: The data that are steadily available from some data store or static probes in network devices. such data can be specified by YANG model.

Custom Data: The data need to be synthesized or processed from raw data from one or more network devices. The data processing function can be statically or dynamically loaded into network devices.

Event-triggered Data: The data are conditionally acquired based on the occurrence of some event. An event can be modeled as a Finite State Machine (FSM).

Streaming Data: The data are continuously or periodically generated.

The above data types are not mutual exclusive. For example, event-triggered data can be simple or custom, and streaming data can be event triggered. The relationships of these data types are illustrated in Figure 1

```
+---------------------------+
| +-----------------------+ |
| | +------------------+  | |
| | | +--------------+ |  | |
| | | | Simple Data  | |  | |
| | | +--------------+ |  | |
| | |    Custom Data   |  | |
| | +------------------+  | |
| | Event-triggered Data  | |
| +-----------------------+ |
|       Streaming Data      |
+---------------------------+
```

Figure 1: Data Type Relationship

Subscription usually deals with event-triggered data and streaming data, and query usually deals with simple data and custom data. It is easy to see that conventional OAM techniques are mostly about querying simple data only. While these techniques are still useful, advanced network telemetry techniques pay more attention on the other three data types, and prefer subscription and custom data query over simple data query.

4.2. Data Objects

Telemetry can be applied on the forwarding plane, the control plane, and the management plane in a network, as well as other sources out of the network, as shown in Figure 2. Therefore, we categorize the network telemetry into four distinct modules.

```
        +------------------------------+
        |                              |
        |      Network Operation       |<-------+
        |         Applications         |        |
        |                              |        |
        +------------------------------+        |
             ^      ^           ^               |
             |      |           |               |
             V      |           V               V
        +-----------|---+--------------+  +-----------+
        |           |   |              |  |           |
        | Control Pl|ane|              |  | External  |
        | Telemetry | <--->            |  | Data and  |
        |           |   |              |  | Event     |
        |      ^    V   |  Management  |  | Telemetry |
        +------|--------+  Plane       |  |           |
        |      V        |  Telemetry   |  +-----------+
        | Forwarding    |              |
        | Plane       <--->            |
        | Telemetry     |              |
        |               |              |
        +---------------+--------------+
```

Figure 2: Layer Category of the Network Telemetry Framework

The rationale of this partition lies in the different telemetry data objects which result in different data source and export locations. Such differences have profound implications on in-network data programming and processing capability, data encoding and transport protocol, and data bandwidth and latency.

We summarize the major differences of the four modules in the following table. Some representative techniques are shown in some table blocks to highlight the technical diversity of these modules.

Module	Control Plane	Management Plane	Forwarding Plane	External Data
Object	control protocol & signaling, RIB, ACL	config. & operation state, MIB	flow & packet QoS, traffic stat., buffer & queue stat.	terminal, social & environ- mental
Export Location	main control CPU, linecard CPU or fwding chip	main control CPU	fwding chip or linecard CPU; main control CPU unlikely	various
Model	YANG, custom	MIB, syslog, YANG, custom	template, YANG, custom	YANG
Encoding	GPB, JSON, XML, plain	GPB, JSON, XML	plain	GPB, JSON XML, plain
Protocol	gRPC,NETCONF, IPFIX,mirror	gPRC,NETCONF,	IPFIX, mirror	gRPC
Transport	HTTP, TCP, UDP	HTTP, TCP	UDP	TCP, UDP

Figure 3: Layer Category of the Network Telemetry Framework

Note that the interaction with the network operation applications can be indirect. For example, in the management plane telemetry, the management plane may need to acquire data from the data plane. Some of the operational states can only be derived from the data plane such as the interface status and statistics. For another example, the control plane telemetry may need to access the FIB in data plane. On the other hand, an application may involve more than one plane simultaneously. For example, an SLA compliance application may require both the data plane telemetry and the control plane telemetry.

4.3. Function Components

At each plane, the telemetry can be further partitioned into five distinct components:

Data Query, Analysis, and Storage: This component works at the application layer. On the one hand, it is responsible for issuing data queries. The queries can be for modeled data through configuration or custom data through programming. The queries can be one shot or subscriptions for events or streaming data. On the other hand, it receives, stores, and processes the returned data from network devices. Data analysis can be interactive to initiate further data queries.

Data Configuration and Subscription: This component deploys data queries on devices. It determines the protocol and channel for applications to acquire desired data. This component is also responsible for configuring the desired data that might not be directly available form data sources. The subscription data can be described by models, templates, or programs.

Data Encoding and Export: This component determines how telemetry data are delivered to the data analysis and storage component. The data encoding and the transport protocol may vary due to the data exporting location.

Data Generation and Processing: The requested data needs to be captured, processed, and formatted in network devices from raw data sources. This may involve in-network computing and processing on either the fast path or the slow path in network devices.

Data Object and Source: This component determines the monitoring object and original data source. The data source usually just provides raw data which needs further processing. A data source can be considered a probe. A probe can be statically installed or dynamically installed.

```
+-----------------------------------------+
|                                         |
|     Data Query, Analysis, & Storage     |
|                                         |
+-----------------------------------------+
        |                   ^
        |                   |
        V                   |
+----------------------+------------------+
| Data Configuration   |                  |
| & Subscription       | Data Encoding    |
| (model, template,    | & Export         |
| & program)           |                  |
+----------------------+------------------|
|                                         |
|          Data Generation                |
|          & Processing                   |
|                                         |
+-----------------------------------------|
|                                         |
|      Data Object and Source             |
|                                         |
+-----------------------------------------+
```

Figure 4: Components in the Network Telemetry Framework

Since most existing standard-related work belongs to the first four components, in the remainder of the document, we focus on these components only.

4.4. Existing Works Mapped in the Framework

The following two tables provide a non-exhaustive list of existing works (mainly published in IETF and with the emphasis on the latest new technologies) and shows their positions in the framework. The details about the mentioned work can be found in Appendix A.

	Query	Subscription
Simple Data	SNMP, NETCONF, YANG, BMP, IOAM, PBT	
Custom Data	DNP, YANG FSM gRPC, NETCONF	
Event-triggered Data		gRPC, NETCONF, YANG PUSH, DNP IOAM, PBT, YANG FSM
Streaming Data		gRPC, NETCONF, IOAM, PBT, DNP IPFIX, IPFPM

Figure 5: Existing Work Mapping I

	Management Plane	Control Plane	Forwarding Plane
data Config. & subscrib.	gRPC, NETCONF, YANG PUSH	NETCONF/YANG	NETCONF/YANG, YANG FSM
data gen. & processing	DNP, YANG	DNP, YANG	In-situ OAM, PBT, IPFPM, DNP
data export	gRPC, NETCONF YANG PUSH	BMP, NETCONF	IPFIX

Figure 6: Existing Work Mapping II

5. Evolution of Network Telemetry

As the network is evolving towards the automated operation, network telemetry also undergoes several levels of evolution.

Level 0 - Static Telemetry: The telemetry data is determined at design time. The network operator can only configure how to use it with limited flexibility.

Level 1 - Dynamic Telemetry: The telemetry data can be dynamically programmed or configured at runtime, allowing a tradeoff among resource, performance, flexibility, and coverage. DNP is an effort towards this direction.

Level 2 - Interactive Telemetry: The network operator can continuously customize the telemetry data in real time to reflect the network operation's visibility requirements. At this level, some tasks can be automated, although ultimately human operators will still need to sit in the middle to make decisions.

Level 3 - Closed-loop Telemetry: Human operators are completely excluded from the control loop. The intelligent network operation engine automatically issues the telemetry data request, analyzes the data, and updates the network operations in closed control loops.

While most of the existing technologies belong to level 0 and level 1, with the help of a clearly defined network telemetry framework, we can assemble the technologies to support level 2 and make solid steps towards level 3.

6. Security Considerations

Given that this document has proposed a framework for network telemetry and the telemetry mechanisms discussed are distinct (in both message frequency and traffic amount) from the conventional network OAM concepts, we must also reflect that various new security considerations may also arise. A number of techniques already exist for securing the data plane, control plane, and the management plane in a network, but the it is important to consider if any new threat vectors are now being enabled via the use of network telemetry procedures and mechanisms.

Security considerations for networks that use telemetry methods may include:

- Telemetry framework trust and policy model;
- Role management and access control for enabling and disabling telemetry capabilities;
- Protocol transport used telemetry data and inherent security capabilities;

- Telemetry data stores, storage encryption and methods of access;

- Tracking telemetry events and any abnormalities that might identify malicious attacks using telemetry interfaces.

Some of the security considerations highlighted above may be minimized or negated with policy management of network telemetry. In a network telemetry deployment it would be advantageous to separate telemetry capabilities into different classes of policies, i.e., Role Based Access Control and Event-Condition-Action policies. Also, potential conflicts between network telemetry mechanisms must be detected accurately and resolved quickly to avoid unnecessary network telemetry traffic propagation escalating into an unintended or intended denial of service attack.

Further discussion and development of this section will be required, and it is expected that this security section, and subsequent policy section will be developed further.

7. IANA Considerations

This document includes no request to IANA.

8. Contributors

The other major contributors of this document are listed as follows.

- Tianran Zhou

- Zhenbin Li

- Daniel King

9. Acknowledgments

We would like to thank Adrian Farrel, Randy Presuhn, Victor Liu, James Guichard, Uri Blumenthal, Giuseppe Fioccola, Yunan Gu, Parviz Yegani, Young Lee, Alexander Clemm, Joe Clarke, and many others who have provided helpful comments and suggestions to improve this document.

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.brockners-inband-oam-requirements]
Brockners, F., Bhandari, S., Dara, S., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mozes, D., Mizrahi, T., Lapukhov, P., and r. remy@barefootnetworks.com, "Requirements for In-situ OAM", draft-brockners-inband-oam-requirements-03 (work in progress), March 2017.

[I-D.fioccola-ippm-multipoint-alt-mark]
Fioccola, G., Cociglio, M., Sapio, A., and R. Sisto, "Multipoint Alternate Marking method for passive and hybrid performance monitoring", draft-fioccola-ippm-multipoint-alt-mark-04 (work in progress), June 2018.

[I-D.ietf-grow-bmp-adj-rib-out]
Evens, T., Bayraktar, S., Lucente, P., Mi, K., and S. Zhuang, "Support for Adj-RIB-Out in BGP Monitoring Protocol (BMP)", draft-ietf-grow-bmp-adj-rib-out-03 (work in progress), December 2018.

[I-D.ietf-grow-bmp-local-rib]
Evens, T., Bayraktar, S., Bhardwaj, M., and P. Lucente, "Support for Local RIB in BGP Monitoring Protocol (BMP)", draft-ietf-grow-bmp-local-rib-02 (work in progress), September 2018.

[I-D.ietf-netconf-udp-pub-channel]
Zheng, G., Zhou, T., and A. Clemm, "UDP based Publication Channel for Streaming Telemetry", draft-ietf-netconf-udp-pub-channel-04 (work in progress), October 2018.

[I-D.ietf-netconf-yang-push]
Clemm, A., Voit, E., Prieto, A., Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "Subscription to YANG Datastores", draft-ietf-netconf-yang-push-22 (work in progress), February 2019.

[I-D.kumar-rtgwg-grpc-protocol]
Kumar, A., Kolhe, J., Ghemawat, S., and L. Ryan, "gRPC Protocol", draft-kumar-rtgwg-grpc-protocol-00 (work in progress), July 2016.

[I-D.openconfig-rtgwg-gnmi-spec]
Shakir, R., Shaikh, A., Borman, P., Hines, M., Lebsack, C., and C. Morrow, "gRPC Network Management Interface (gNMI)", draft-openconfig-rtgwg-gnmi-spec-01 (work in progress), March 2018.

[I-D.pedro-nmrg-anticipated-adaptation]
Martinez-Julia, P., "Exploiting External Event Detectors to Anticipate Resource Requirements for the Elastic Adaptation of SDN/NFV Systems", draft-pedro-nmrg-anticipated-adaptation-02 (work in progress), June 2018.

[I-D.song-ippm-postcard-based-telemetry]
Song, H., Zhou, T., Li, Z., and J. Shin, "Postcard-based In-band Flow Data Telemetry", draft-song-ippm-postcard-based-telemetry-01 (work in progress), December 2018.

[I-D.song-opsawg-dnp4iq]
Song, H. and J. Gong, "Requirements for Interactive Query with Dynamic Network Probes", draft-song-opsawg-dnp4iq-01 (work in progress), June 2017.

[I-D.zhou-netconf-multi-stream-originators]
Zhou, T., Zheng, G., Voit, E., Clemm, A., and A. Bierman, "Subscription to Multiple Stream Originators", draft-zhou-netconf-multi-stream-originators-03 (work in progress), October 2018.

[RFC1157] Case, J., Fedor, M., Schoffstall, M., and J. Davin, "Simple Network Management Protocol (SNMP)", RFC 1157, DOI 10.17487/RFC1157, May 1990, <https://www.rfc-editor.org/info/rfc1157>.

[RFC2981] Kavasseri, R., Ed., "Event MIB", RFC 2981, DOI 10.17487/RFC2981, October 2000, <https://www.rfc-editor.org/info/rfc2981>.

[RFC3416] Presuhn, R., Ed., "Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3416, DOI 10.17487/RFC3416, December 2002, <https://www.rfc-editor.org/info/rfc3416>.

[RFC3877] Chisholm, S. and D. Romascanu, "Alarm Management Information Base (MIB)", RFC 3877, DOI 10.17487/RFC3877, September 2004, <https://www.rfc-editor.org/info/rfc3877>.

[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>.

[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>.

[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>.

[RFC7276] Mizrahi, T., Sprecher, N., Bellagamba, E., and Y. Weingarten, "An Overview of Operations, Administration, and Maintenance (OAM) Tools", RFC 7276, DOI 10.17487/RFC7276, June 2014, <https://www.rfc-editor.org/info/rfc7276>.

[RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <https://www.rfc-editor.org/info/rfc7540>.

[RFC7799] Morton, A., "Active and Passive Metrics and Methods (with Hybrid Types In-Between)", RFC 7799, DOI 10.17487/RFC7799, May 2016, <https://www.rfc-editor.org/info/rfc7799>.

[RFC7854] Scudder, J., Ed., Fernando, R., and S. Stuart, "BGP Monitoring Protocol (BMP)", RFC 7854, DOI 10.17487/RFC7854, June 2016, <https://www.rfc-editor.org/info/rfc7854>.

[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>.

Appendix A. A Survey on Existing Network Telemetry Techniques

We provide an overview of the challenges and existing solutions for each network telemetry module.

A.1. Management Plane Telemetry

A.1.1. Requirements and Challenges

The management plane of the network element interacts with the Network Management System (NMS), and provides information such as performance data, network logging data, network warning and defects data, and network statistics and state data. Some legacy protocols are widely used for the management plane, such as SNMP and Syslog. However, these protocols are insufficient to meet the requirements of the automatic network operation applications.

New management plane telemetry protocols should consider the following requirements:

Convenient Data Subscription: An application should have the freedom to choose the data export means such as the data types and the export frequency.

Structured Data: For automatic network operation, machines will replace human for network data comprehension. The schema languages such as YANG can efficiently describe structured data and normalize data encoding and transformation.

High Speed Data Transport: In order to retain the information, a server needs to send a large amount of data at high frequency. Compact encoding formats are needed to compress the data and improve the data transport efficiency. The push mode, by replacing the poll mode, can also reduce the interactions between clients and servers, which help to improve the server's efficiency.

A.1.2. Push Extensions for NETCONF

NETCONF [RFC6241] is one popular network management protocol, which is also recommended by IETF. Although it can be used for data collection, NETCONF is good at configurations. YANG Push

[I-D.ietf-netconf-yang-push] extends NETCONF and enables subscriber applications to request a continuous, customized stream of updates from a YANG datastore. Providing such visibility into changes made upon YANG configuration and operational objects enables new capabilities based on the remote mirroring of configuration and operational state. Moreover, distributed data collection mechanism [I-D.zhou-netconf-multi-stream-originators] via UDP based publication channel [I-D.ietf-netconf-udp-pub-channel] provides enhanced efficiency for the NETCONF based telemetry.

A.1.3. gRPC Network Management Interface

gRPC Network Management Interface (gNMI) [I-D.openconfig-rtgwg-gnmi-spec] is a network management protocol based on the gRPC [I-D.kumar-rtgwg-grpc-protocol] RPC (Remote Procedure Call) framework. With a single gRPC service definition, both configuration and telemetry can be covered. gRPC is an HTTP/2 [RFC7540] based open source micro service communication framework. It provides a number of capabilities which are well-suited for network telemetry, including:

- Full-duplex streaming transport model combined with a binary encoding mechanism provided further improved telemetry efficiency.

- gRPC provides higher-level features consistency across platforms that common HTTP/2 libraries typically do not. This characteristic is especially valuable for the fact that telemetry data collectors normally reside on a large variety of platforms.

- The built-in load-balancing and failover mechanism.

A.2. Control Plane Telemetry

A.2.1. Requirements and Challenges

The control plane telemetry refers to the health condition monitoring of different network protocols, which covers Layer 2 to Layer 7. Keeping track of the running status of these protocols is beneficial for detecting, localizing, and even predicting various network issues, as well as network optimization, in real-time and in fine granularity.

One of the most challenging problems for the control plane telemetry is how to correlate the E2E Key Performance Indicators (KPI) to a specific layer's KPIs. For example, an IPTV user may describe his User Experience (UE) by the video fluency and definition. Then in case of an unusually poor UE KPI or a service disconnection, it is non-trivial work to delimit and localize the issue to the responsible protocol layer (e.g., the Transport Layer or the Network Layer), the responsible protocol (e.g., ISIS or BGP at the Network Layer), and finally the responsible device(s) with specific reasons.

Traditional OAM-based approaches for control plane KPI measurement include PING (L3), Tracert (L3), Y.1731 (L2) and so on. One common issue behind these methods is that they only measure the KPIs instead of reflecting the actual running status of these protocols, making them less effective or efficient for control plane troubleshooting and network optimization. An example of the control plane telemetry is the BGP monitoring protocol (BMP), it is currently used to monitoring the BGP routes and enables rich applications, such as BGP peer analysis, AS analysis, prefix analysis, security analysis, and so on. However, the monitoring of other layers, protocols and the cross-layer, cross-protocol KPI correlations are still in their infancy (e.g., the IGP monitoring is missing), which require substantial further research.

A.2.2. BGP Monitoring Protocol

BGP Monitoring Protocol (BMP) [RFC7854] is used to monitor BGP sessions and intended to provide a convenient interface for obtaining route views.

The BGP routing information is collected from the monitored device(s) to the BMP monitoring station by setting up the BMP TCP session. The BGP peers are monitored by the BMP Peer Up and Peer Down Notifications. The BGP routes (including Adjacency_RIB_In [RFC7854], Adjacency_RIB_out [I-D.ietf-grow-bmp-adj-rib-out], and Local_Rib [I-D.ietf-grow-bmp-local-rib] are encapsulated in the BMP Route Monitoring Message and the BMP Route Mirroring Message, in the form of both initial table dump and real-time route update. In addition, BGP statistics are reported through the BMP Stats Report Message, which could be either timer triggered or event-driven. More BMP extensions can be explored to enrich the applications of BGP monitoring.

A.3. Data Plane Telemetry

A.3.1. Requirements and Challenges

An effective data plane telemetry system relies on the data that the network device can expose. The data’s quality, quantity, and timeliness must meet some stringent requirements. This raises some challenges to the network data plane devices where the first hand data originate.

- o A data plane device's main function is user traffic processing and forwarding. While supporting network visibility is important, the telemetry is just an auxiliary function, and it should not impede normal traffic processing and forwarding (i.e., the performance is not lowered and the behavior is not altered due to the telemetry functions).

- o The network operation applications requires end-to-end visibility from various sources, which results in a huge volume of data. However, the sheer data quantity should not stress the network bandwidth, regardless of the data delivery approach (i.e., through in-band or out-of-band channels).

- o The data plane devices must provide timely data with the minimum possible delay. Long processing, transport, storage, and analysis delay can impact the effectiveness of the control loop and even render the data useless.

- o The data should be structured and labeled, and easy for applications to parse and consume. At the same time, the data types needed by applications can vary significantly. The data plane devices need to provide enough flexibility and programmability to support the precise data provision for applications.

- o The data plane telemetry should support incremental deployment and work even though some devices are unaware of the system. This challenge is highly relevant to the standards and legacy networks.

The industry has agreed that the data plane programmability is essential to support network telemetry. Newer data plane chips are all equipped with advanced telemetry features and provide flexibility to support customized telemetry functions.

A.3.2. Technique Taxonomy

There can be multiple possible dimensions to classify the data plane telemetry techniques.

Active and Passive: The active and passive methods (as well as the hybrid types) are well documented in [RFC7799]. The passive methods include TCPDUMP, IPFIX [RFC7011], sflow, and traffic mirror. These methods usually have low data coverage. The bandwidth cost is very high in order to improve the data coverage. On the other hand, the active methods include Ping, Traceroute, OWAMP [RFC4656], and TWAMP [RFC5357]. These methods are intrusive and only provide indirect network measurement results. The hybrid methods, including in-situ OAM

[I-D.brockners-inband-oam-requirements], IPFPM [RFC8321], and Multipoint Alternate Marking [I-D.fioccola-ippm-multipoint-alt-mark], provide a well-balanced and more flexible approach. However, these methods are also more complex to implement.

In-Band and Out-of-Band: The telemetry data, before being exported to some collector, can be carried in user packets. Such methods are considered in-band (e.g., in-situ OAM [I-D.brockners-inband-oam-requirements]). If the telemetry data is directly exported to some collector without modifying the user packets, Such methods are considered out-of-band (e.g., postcard-based INT). It is possible to have hybrid methods. For example, only the telemetry instruction or partial data is carried by user packets (e.g., IPFPM [RFC8321]).

E2E and In-Network: Some E2E methods start from and end at the network end hosts (e.g., Ping). The other methods work in networks and are transparent to end hosts. However, if needed, the in-network methods can be easily extended into end hosts.

Flow, Path, and Node: Depending on the telemetry objective, the methods can be flow-based (e.g., in-situ OAM [I-D.brockners-inband-oam-requirements]), path-based (e.g., Traceroute), and node-based (e.g., IPFIX [RFC7011]).

A.3.3. The IPFPM technology

The Alternate Marking method is efficient to perform packet loss, delay, and jitter measurements both in an IP and Overlay Networks, as presented in IPFPM [RFC8321] and [I-D.fioccola-ippm-multipoint-alt-mark].

This technique can be applied to point-to-point and multipoint-to-multipoint flows. Alternate Marking creates batches of packets by alternating the value of 1 bit (or a label) of the packet header. These batches of packets are unambiguously recognized over the network and the comparison of packet counters for each batch allows the packet loss calculation. The same idea can be applied to delay measurement by selecting ad hoc packets with a marking bit dedicated for delay measurements.

Alternate Marking method needs two counters each marking period for each flow under monitor. For instance, by considering n measurement points and m monitored flows, the order of magnitude of the packet counters for each time interval is n*m*2 (1 per color).

Since networks offer rich sets of network performance measurement data (e.g packet counters), traditional approaches run into limitations. One reason is the fact that the bottleneck is the generation and export of the data and the amount of data that can be reasonably collected from the network. In addition, management tasks related to determining and configuring which data to generate lead to significant deployment challenges.

Multipoint Alternate Marking approach, described in [I-D.fioccola-ippm-multipoint-alt-mark], aims to resolve this issue and makes the performance monitoring more flexible in case a detailed analysis is not needed.

An application orchestrates network performance measurements tasks across the network to allow an optimized monitoring and it can calibrate how deep can be obtained monitoring data from the network by configuring measurement points roughly or meticulously.

Using Alternate Marking, it is possible to monitor a Multipoint Network without examining in depth by using the Network Clustering (subnetworks that are portions of the entire network that preserve the same property of the entire network, called clusters). So in case there is packet loss or the delay is too high the filtering criteria could be specified more in order to perform a detailed analysis by using a different combination of clusters up to a per-flow measurement as described in IPFPM [RFC8321].

In summary, an application can configure end-to-end network monitoring. If the network does not experiment issues, this approximate monitoring is good enough and is very cheap in terms of network resources. However, in case of problems, the application becomes aware of the issues from this approximate monitoring and, in order to localize the portion of the network that has issues, configures the measurement points more exhaustively. So a new detailed monitoring is performed. After the detection and resolution of the problem the initial approximate monitoring can be used again.

A.3.4. Dynamic Network Probe

Hardware-based Dynamic Network Probe (DNP) [I-D.song-opsawg-dnp4iq] provides a programmable means to customize the data that an application collects from the data plane. A direct benefit of DNP is the reduction of the exported data. A full DNP solution covers several components including data source, data subscription, and data generation. The data subscription needs to define the custom data which can be composed and derived from the raw data sources. The data generation takes advantage of the moderate in-network computing to produce the desired data.

While DNP can introduce unforeseeable flexibility to the data plane telemetry, it also faces some challenges. It requires a flexible data plane that can be dynamically reprogrammed at run-time. The programming API is yet to be defined.

A.3.5. IP Flow Information Export (IPFIX) protocol

Traffic on a network can be seen as a set of flows passing through network elements. IP Flow Information Export (IPFIX) [RFC7011] provides a means of transmitting traffic flow information for administrative or other purposes. A typical IPFIX enabled system includes a pool of Metering Processes collects data packets at one or more Observation Points, optionally filters them and aggregates information about these packets. An Exporter then gathers each of the Observation Points together into an Observation Domain and sends this information via the IPFIX protocol to a Collector.

A.3.6. In-Situ OAM

Traditional passive and active monitoring and measurement techniques are either inaccurate or resource-consuming. It is preferable to directly acquire data associated with a flow's packets when the packets pass through a network. In-situ OAM (iOAM) [I-D.brockners-inband-oam-requirements], a data generation technique, embeds a new instruction header to user packets and the instruction directs the network nodes to add the requested data to the packets. Thus, at the path end, the packet's experience gained on the entire forwarding path can be collected. Such firsthand data is invaluable to many network OAM applications.

However, iOAM also faces some challenges. The issues on performance impact, security, scalability and overhead limits, encapsulation difficulties in some protocols, and cross-domain deployment need to be addressed.

A.3.7. Postcard Based Telemetry

PBT [I-D.song-ippm-postcard-based-telemetry] is an alternative to IOAM. PBT directly exports data at each node through an independent packet. PBT solves several issues of IOAM. It can also help to identify packet drop location in case a packet is dropped on its forwarding path.

A.4. External Data and Event Telemetry

Events that occur outside the boundaries of the network system are another important source of telemetry information. Correlating both internal telemetry data and external events with the requirements of network systems, as presented in Exploiting External Event Detectors to Anticipate Resource Requirements for the Elastic Adaptation of SDN/NFV Systems [I-D.pedro-nmrg-anticipated-adaptation], provides a strategic and functional advantage to management operations.

A.4.1. Requirements and Challenges

As with other sources of telemetry information, the data and events must meet strict requirements, especially in terms of timeliness, which is essential to properly incorporate external event information to management cycles. Thus, the specific challenges are described as follows:

- The role of external event detector can be played by multiple elements, including hardware (e.g. physical sensors, such as seismometers) and software (e.g. Big Data sources that analyze streams of information, such as Twitter messages). Thus, the transmitted data must support different shapes but, at the same time, follow a common but extensible ontology.

- Since the main function of the external event detectors is to perform the notifications, their timeliness is assumed. However, once messages have been dispatched, they must be quickly collected and inserted into the control plane with variable priority, which will be high for important sources and/or important events and low for secondary ones.

- The ontology used by external detectors must be easily adopted by current and future devices and applications. Therefore, it must be easily mapped to current information models, such as in terms of YANG.

Organizing together both internal and external telemetry information will be key for the general exploitation of the management possibilities of current and future network systems, as reflected in the incorporation of cognitive capabilities to new hardware and software (virtual) elements.

A.4.2. Sources of External Events

To ensure that the information provided by external event detectors and used by the network management solutions is meaningful for the management purposes, the network telemetry framework must ensure that such detectors (sources) are easily connected to the management solutions (sinks). This requires the specification of a simple taxonomy of detectors and match it to the connectors and/or interfaces required to connect them.

Once detectors are classified in such taxonomy, their definitions are enlarged with the qualities and other aspects used to handle them and represented in the ontology and information model (e.g. YANG). Therefore, differentiating several types of detectors as potential sources of external events is essential for the integrity of the management framework. We thus differentiate the following source types of external events:

o Smart objects and sensors. With the consolidation of the Internet of Things˜(IoT) any network system will have many smart objects attached to its physical surroundings and logical operation environments. Most of these objects will be essentially based on sensors of many kinds (e.g. temperature, humidity, presence) and the information they provide can be very useful for the management of the network, even when they are not specifically deployed for such purpose. Elements of this source type will usually provide a specific protocol for interaction, especially one of those protocols related to IoT, such as the Constrained Application Protocol (CoAP). It will be used by the telemetry framework to interact with the relevant objects.

o Online news reporters. Several online news services have the ability to provide enormous quantity of information about different events occurring in the world. Some of those events can impact on the network system managed by a specific framework and, therefore, it will be interested on getting such information. For instance, diverse security reports, such as the Common Vulnerabilities and Exposures (CVE), can be issued by the corresponding authority and used by the management solution to update the managed system if needed. Instead of a specific protocol and data format, the sources of this kind of information usually follow a relaxed but structured format. This format will be part of both the ontology and information model of the telemetry framework.

o Global event analyzers. The advance of Big Data analyzers provides a huge amount of information and, more interestingly, the identification of events detected by analyzing many data streams from different origins. In contrast with the other types of sources, which are focused in specific events, the detectors of this source type will detect very generic events. For example, a sports event takes place and some unexpected movement makes it highly interesting and many people connects to sites that are covering such event. The systems supporting the services that cover the event can be affected by such situation so their management solutions should be aware of it. In contrast with the other source types, a new information model, format, and reporting protocol is required to integrate the detectors of this type with the management solution.

Additional types of detector types can be added to the system but they will be generally the result of composing the properties offered by these main classes. In any case, future revisions of the network telemetry framework will include the required types that cover new circumstances and that cannot be obtained by composition.

A.4.3. Connectors and Interfaces

For allowing external event detectors to be properly integrated with other management solutions, both elements must expose interfaces and protocols that are subject to their particular objective. Since external event detectors will be focused on providing their information to their main consumers, which generally will not be limited to the network management solutions, the framework must include the definition of the required connectors for ensuring the interconnection between detectors (sources) and their consumers within the management systems (sinks) are effective.

In some situations, the interconnection between the external event detectors and the management system is via the management plane. For those situations there will be a special connector that provides the typical interfaces found in most other elements connected to the management plane. For instance, the interfaces will accomplish with a specific information model (YANG) and specific telemetry protocol, such as NETCONF, SNMP, or gRPC.

Authors' Addresses

Haoyu Song (editor)
Huawei
2330 Central Expressway
Santa Clara
USA

Email: haoyu.song@huawei.com

Zhenqiang Li
China Mobile
No. 32 Xuanwumenxi Ave., Xicheng District
Beijing, 100032
P.R. China

Email: lizhenqiang@chinamobile.com

Pedro Martinez-Julia
NICT
4-2-1, Nukui-Kitamachi
Koganei, Tokyo 184-8795
Japan

Email: pedro@nict.go.jp

Laurent Ciavaglia
Nokia
Villarceaux 91460
France

Email: laurent.ciavaglia@nokia.com

Aijun Wang
China Telecom
Beiqijia Town, Changping District
Beijing, 102209
P.R. China

Email: wangaj.bri@chinatelecom.cn

Operations and Management Area Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 4, 2020

Q. Sun
H. Xu
China Telecom
B. Wu, Ed.
Q. Wu, Ed.
Huawei
C. Eckel, Ed.
Cisco Systems

July 3, 2019

A YANG Data Model for SD-WAN Service Delivery
draft-sun-opsawg-sdwan-service-model-04

Abstract

This document provides a YANG data model for an SD-WAN service. An SD-WAN service is a connectivity service offered by a service provider network to provide connectivity across different locations of a customer network or between a customer network and an external network, such as the Internet or a private/public cloud network. This connectivity is provided as an overlay constructed using one of more underlay networks. The model can be used by a service orchestrator of a service provider to request, configure, and manage the components of an SD-WAN service.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

An SD-WAN service is a connectivity service offered by a service provider network to provide connectivity across different locations of a customer network or between a customer network and an external network. Compared to a conventional PE-based connectivity service as defined in Layer 3 VPN Service Model [RFC8299] and Layer 2 VPN Service Model [RFC8466], an SD-WAN service is a CE-based connectivity service that uses the Internet or PE-based connectivity services as underlay connectivity services. More specially, an SD-WAN service is an overlay connectivity service that provides the flexibility of adding, removing, or moving services without needing to change the underlay networks.

Besides being an overlay service, an SD-WAN Service has the following characteristics:

- o Hybrid WAN access: The CE could connect to a variety of Internet access technologies, including fiber, cable, DSL-based, WiFi, or 4G/Long Term Evolution (LTE), which implies wider reachability and shorter provisioning cycles. It can also use private VPN connectivity services defined in [RFC4364] and [RFC4664], or Operator Ethernet Services, as defined in [MEF51.1], to take advantage of better performance.

- o Application based traffic forwarding: There are diverse applications used in enterprises, such as VoIP calling, video conferencing, streaming media, etc. Application traffic across the WAN will be forwarded based on business priorities, SLA requirements, or other enterprise requirements.

- o Centralized service management: Subscribers of the service need to be provided a single point (such as a web portal) from which to dynamically add or modify services, such as configuring application policies, adding new sites, or adding new underlay connectivity services.

This draft specifies the SD-WAN service YANG model which is modelled from a customer perspective. The model parameters can be used as an input to automated control and configuration applications to manage SD-WAN services.

1.1. Terminology

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 RFC2119 [RFC2119].

1.2. Definitions

CE Device: Customer Edge Device , as per Provider Provisioned VPN Terminology [RFC4026] .

CE-based VPN: Refers to Provider Provisioned VPN Terminology [RFC4026]

PE Device: Provider Edge Device, as per Provider Provisioned VPN Terminology [RFC4026]

PE-Based VPNs: Refers to Provider Provisioned VPN Terminology [RFC4026]

SD-WAN: An automated, programmatic approach to managing enterprise network connectivity and circuit usage. It extends software-defined networking (SDN) into an application that businesses can use to quickly create a hybrid WAN, which comprises business-grade IP VPN, broadband Internet, and wireless services or multiple WANs of the same or different types. SD-WAN is also deemed as extended CE-based VPN.

SD-WAN Controller: Refers to the abstract entity that combines Control Plane (CP) and Management Plane (MP) defined in SDN: Layers and Architecture Terminology [RFC7426], to configure, manage and control the CEs and other corresponding SD-WAN components.

Underlay network: A network that provides connectivity across SD-WAN sites and over which customer network packets are tunnelled. An underlay network does not need to be aware that it is carrying overlay customer network packets. Addresses on an underlay network appear as "outer addresses" in encapsulated overlay packets. In general, an underlay network can use a completely different protocol (and address family) from that of the overlay network.

Overlay network: A virtual network in which the separation of customer networks is hidden from the underlying physical infrastructure. That is, the underlying transport networks do not need to know about customer separation to correctly forward traffic. IPsec tunnels [RFC6071] are an example of an L3 overlay network.

2. High Level Overview of SD-WAN Service

From a customer perspective, an example of SD-WAN service network is shown in figure 1.

```
                           +-------------+
      +------------+       |    +---+    |
      | Controller +----+  |    |CN |    |   Legend:Customer Network
      +------------+    |  |    +---+    |
                        |  |      | site3|
                        |  |   +--+--+   |
                        +--|---|CE 4 |   |
                        |  |   +--+--+   |
                        |  +-------------+
                        |         |
                        +------------------ ----+
                        |        -----          |
           +--------------+   /  MPLS  \   +-----------------+
           |            | |  |   WAN    |__|    |            |
           |            | |  /\        /\  \ +--+--+         |
           |            | | /   +-----+  \ |\|CE 1 +-+       |
           | +---+ +----++|/              \|/+--+--+ |  +---+|
           | |CN +--+ CE 3||              \          +--+CN ||
           | +---+ +-----+|      ------   /|\+--+--+ |  +---+|
           |            | |\  /Internet\ / |/|CE 2 +-+       |
           |            | | --|   WAN   |__/ +--+--+         |
           |      site 2| |    \       /   |  site 1         |
           +--------------+      ------    +-----------------+
                        |           |
                        |    +-------------+
                        |    |   +----+    |
                        +----|---+ CE5|    |
                             |   +----+    |
                             |site 4|      |
                             |      |      |
                             |    +---+    |
                             |    |CN |    |
                             |    +---+    |
                             +-------------+
```

figure 1 SD-WAN network example

As shown in figure 1, the SD-WAN network consists of a number of sites, which are connected through Internet or MPLS VPN.

Within each site, a CE is connected with customer's network on one side, and is also connected to Internet, or to private WAN, or to both on the other side. The customer network could be an L2 or L3 network. For the WAN side, Internet provides ubiquitous IP connectivity via access network like Broadband access or LTE access, while MPLS WAN, like conventional VPN, provides secure and committed connectivity. The boundary between the customer and the service provider is between customer node and the CE device.

Additionally, a site could deploy one or more CEs to improve availability.

The controller is a centralized entity that manages all the CEs involved in the SD-WAN. The controller could provide bootstrapping of the CEs, ongoing CE configuration, and establishment of secured tunnels between CEs to support the SD-WAN service and application policy enforcement. Various IP tunnelling options (e.g., GRE [RFC2784] and IPSec [RFC6071]), could be used depending on whether traffic from the site is across underlying private VPN or public Internet, and the specific definition is out of scope of this document.

Besides basic connectivity between the sites, the SD-WAN service could be extended by providing direct Internet connectivity, cloud network connectivity, or conventional MPLS VPN interoperability.

3. Service Data Model Usage

The SD-WAN service model provides an abstracted interface to request, configure, and manage the components of an SD-WAN service.

A typical usage for this model is as an input to a service orchestrator that is responsible for service management. Based on the user's service request, the service orchestrator can instruct the SD-WAN controller to add a new site,VPN or application policy in real-time. The orchestrator could orchestrator the other network, such as legacy MPLS VPN network to interconnect with SD-WAN network where Layer 2 VPN Service Mode [RFC8466] or Layer 3 VPN Service Model [RFC8299] could be used.

```
                         ----------------------------
                        | Customer Service Requester |
                         ----------------------------
                                      |
                             SD-WAN   |
                             Service  |
                             Model    |
                                      |
                           -------------------------
                          | Service Orchestrator    |
                           -----------+---------+---
                                      |         |
                 ----------------------+-   ----+----
                |   SD-WAN  Controller |  |  NMS    |
                 --------*------------*-    ----*----
                        /              \       /
                       /                \    /
                      /                  \ /
     ----------------/-  -----------------X  ------------------------
                    /                    / \
                   /           ---     /    \
                  /           /   \/         \
           ++++++++          |  MPLS |        \++++++++
           + CE A +           \  VPN/          + CE B +
           ++++++++            \---/           ++++++++
                               /---\
           Site A             /     \            Site B
                             |Internet
                              \     /
                               \---/
```

Reference Architecture for the Use of SD-WAN Service Model Usage

For an SD-WAN to be established under the SP's control, the customer informs the Service Provider of which sites should become part of the requested service and what types of policy will provide. And then the SP configures and updates the service base on the service model and the available resources derived from the SD-WAN controller, and then provisions and manages the customer's service through the SD-WAN controller. How the SD-WAN controller to control and manage the CEs is out of scope of the document.

4. Design of the Data Model

An SD-WAN service consist of two service components:

1. SD-WAN connectivity service

2. SD-WAN application policy service

4.1. SD-WAN connectivity service

SD-WAN connectivity service is the basic component of the SD-WAN service that represents a virtual connection between two or more customer sites. In this model, each virtual connection is defined as a VPN. Each customer can have one or more VPNs, and each VPN can be established between a subset of sites. The association of sites and VPNs is modelled by VPN endpoints.

4.1.1. VPNs

The "sdwan-vpn" list item contains service parameters that apply to an SD-WAN VPN. These parameters are specified as follows:

- o The "vpn-id" leaf is under the vpn-service list, and providers a unique ID for a VPN.

- o The "endpoints" list is under the vpn-service list. Each "endpoint" is a logical point associated with a site. The two main functions of the endpoint are the association of a VPN with a site and per site application based policy enforcement.

- o The "topology" leaf is under the vpn-service list, which refers to a specific topology of the VPN service. Different VPN connection topology can be used. For a VPN with a few sites, simple topologies such as hub-and-spoke or full-mesh can be used. For a large VPN, a hierarchical topology may be taken.

- o The "performance-objectives" container specifies the performance-related properties of an SD-WAN VPN that can be measured. System uptime is the only performance objective defined currently. It indicates the proportion of time, during a given time period that the service is working from the customer perspective. Three parameters are defined, including the start time of the evaluation, the time interval of the evaluation, and the service uptime defined by a percentage.

- o The "reserved-prefixes" container specifies the IP Prefixes that need to be reserved for Service Provider management purposes, such as diagnostics, so as to ensure they are not overlapping with IP Prefixes used by the customer network.

```
                                 ------
                               / MPLS   \
                              |   VPN    |
    +----------------+          \       /   +----------------+
----+    ---         |   VPN1    ------     +      ---       +------
    |   |EP1+--------+----------------------+-----+EP1|      |
    |    ---         |                      |      ---       |
    |    ---         |   VPN2               |      ---       |
----+   |EP2+--------+----------------------+-----+EP2|      +------
    |    ---  Site 1 |          ------      |      ---  Site2|
    +-- -------------+        /        \    +----------------+
                             | Internet |
                              \        /
                                ------
```

figure 3 SD-WAN VPN example

4.1.2. Sites

A site represents a customer office located at a specific geographic location. The "sites" container specifies the following parameters:

- o "site-id: uniquely identifies the site within the overall network infrastructure.

- o "device" specifies the device type (physical or virtual device) and the number of the devices.

- o "lan-accesses": Specifies the customer network access link parameters. A "site" is composed of at least one "lan-access" where one or more subnets can reside.The "lan-access" consists of the following categories of parameters:

  * "bearer": defines requirements of the attachment (below Layer 3), bearer type including Ethernet, etc.

  * IP Connection: defines Layer 3 parameters of the attachment, including IPv4 connection parameters and IPv6 connection parameters.

- o "wan-accesses": Specifies the WAN access link parameters. A "site" is composed of at least one "wan-access". The WAN access can be further specified by access type, service provider name, and bandwidth of the WAN connectivity. The "wan-access" consists of the following categories of parameters:

  * "access-type":specifies whether the access is Broadband Internet, Wireless Internet or private circuit.

* "access-provider": specifies the service provider name.

* bandwidth: specifies the WAN link bandwidth including input and output bandwidth.

* "bearer": defines requirements of the attachment (below Layer 3), bearer type including Ethernet, etc.

* IP Connection: defines Layer 3 parameters of the attachment, including IPv4 connection parameters and IPv6 connection parameters.

```
                    +---------------------------------+
                    |              site               |
                    |    |   |            |     |     |
                    |    |   |            |     |     |
                    |   LAN1 LAN2         LAN3 LAN4   |
                    |  +--------+       +--------+    |
                    |  |        |       |        |    |
                    |  |Device 1|       |Device 2|    |
                    |  +---+----+       +----+---+    |
                    |  WAN1|  WAN2     WAN3  | WAN4   |
                    |      |    \       /    |        |
                    +------+-----------------+--------+
                           |      \   /      |
                           |       \ /       |
                         -----      /\     -----
                       /        \  /  \  /        \
                      | MPLS VPN |-    -| Internet |
                       \        /        \        /
                         -----              -----
```

figure 4 Site example

4.2. Application based Policy Service

The connectivity service establishes a virtual connection for the enterprise network, and the Application based Policy Service is designed to ensure business-critical and real-time application experience while also ensuring the security and corporate policies.

Typically, application policies common to each VPN can be defined and then enforced when traffic from a customer's network at a particular site is sent over the WAN.

The application policy assignment is defined under the VPN endpoint container to specify the mapping of application flow name or application group name and their associated policy list names. If an application flow and the application flow group in which the Application Flow is a member are both assigned a policy at an VPN End Point, the policy assigned to the application flow will supersedes the group policy.

The application policy per VPN consist of three lists under the VPN container:

- o application flow list: Describes the characteristics of an enterprise application and is used to identify applications, e.g., based on layer 3 source and destination addresses, layer 4 ports, layer 4 protocol, etc.

- o application group list: Describes application flow aggregation, which is used to deliver aggregation policies, such as bandwidth restrictions for a group of applications.

- o policy list: Defines the application's policy set. Since SD-WAN has more than one WAN connectivity and various encrypted or unencrypted overlay tunnels, there could be multiple tunnel or link selection combination. In this model, different path selection policies are combined to meet different needs based on application SLA, security, cost, and so on. For example, when different applications in a branch need to pass over the WAN, according to the application-aware policy requirements and the IP forwarding table, the Internet application or the SaaS application can be accessed through the Internet, and the data center FTP application can use the Internet encrypted tunnel as the primary path, and the tunnel could only be over broadband Internet instead of wireless internet. This policy combination is not an exhaustive list and could be augmented according to business needs.

An example of a classification of application flows is as follows:

The HTTP traffic from the 192.0.2.0/24 LAN destined for port 80 will be classified in app-id 1.

The FTP traffic from the 192.0.2.0/24 LAN destined for 203.0.113.1/32 will be classified in app-id 2.

An example of a policy list is as follows:

```
      "policy": [
        {
          "policy-id": "pol-a",
          "policy-package":
               {
                   "encryption": "false",
                   "internet-breakout": "true"
                   "public-private": "public",
                   "billing-method": "flat-only"
                   "backup": "false",
                   "bandwidth": "20","50"
                }
         },
        {
          "policy-id": "pol-b",
          "policy-package":
            {
               "encryption": "true",
               "internet-breakout": "false"
                "public-private": "public",
                "billing-method": "flat-only"
                "backup": "false",
                "bandwidth": "50","none"
                }
        }
      ]
```


An example of an application policy list is as follows:


```
      "app-policy": [
        {
          "app-id": "1"
          "policy-id": "pol-a",
         },
        {
           "app-id": "1"
           "policy-id": "pol-b",
        }
      ]
```


5. Modules Tree Structure

This document defines an SD-WAN service YANG data model.

```
module: ietf-sdwan-svc
  +--rw sdwan-svc
     +--rw vpn-services
     |  +--rw vpn-service* [vpn-id]
```

```
        |     +--rw vpn-id                   svc-id
        |     +--rw topology?                identityref
        |     +--rw performance-objective
        |     |  +--rw start-time?         yang:date-and-time
        |     |  +--rw duration?           string
        |     |  +--rw uptime-objective
        |     |     +--rw duration?   decimal64
        |     +--rw reserved-prefixes
        |     |  +--rw prefix*   inet:ip-prefix
        |     +--rw application* [app-id]
        |     |  +--rw app-id    svc-id
        |     |  +--rw ac* [name]
        |     |     +--rw name                        string
        |     |     +--rw (match-type)?
        |     |        +--:(match-flow)
        |     |        |  +--rw match-flow
        |     |        |     +--rw ethertype?         uint16
        |     |        |     +--rw cvlan?             uint8
        |     |        |     +--rw ipv4-src-prefix?   inet:ipv4-prefix
        |     |        |     +--rw ipv4-dst-prefix?   inet:ipv4-prefix
        |     |        |     +--rw l4-src-port?       inet:port-number
        |     |        |     +--rw l4-dst-port?       inet:port-number
        |     |        |     +--rw ipv6-src-prefix?   inet:ipv6-prefix
        |     |        |     +--rw ipv6-dst-prefix?   inet:ipv6-prefix
        |     |        |     +--rw protocol-field?    union
        |     |        +--:(match-application)
        |     |           +--rw match-application?   identityref
        |     +--rw application-group* [app-group-id]
        |     |  +--rw app-group-id    svc-id
        |     |  +--rw app-id*         -> ../../application/app-id
        |     +--rw policy* [policy-id]
        |     |  +--rw policy-id         svc-id
        |     |  +--rw policy-package
        |     |     +--rw encryption?       enumeration
        |     |     +--rw public-private?   enumeration
        |     |     +--rw local-breakout?   boolean
        |     |     +--rw billing-method?   enumeration
        |     |     +--rw backup-path?      enumeration
        |     |     +--rw bandwidth
        |     |        +--rw commit?   uint32
        |     |        +--rw max?      uint32
        |     +--rw endpoints* [endpoint-id]
        |        +--rw endpoint-id           svc-id
        |        +--rw site-role?            identityref
        |        +--rw site-attachment
        |        |  +--rw site-id?   -> /sdwan-svc/sites/site/site-id
        |        +--rw endpoint-policy-map
        |           +--rw app-group-policy* [app-group-id]
```

```
        |           |  +--rw app-group-id    leafref
        |           |  +--rw policy-id?      leafref
        |           +--rw app-policy* [app-id]
        |              +--rw app-id       leafref
        |              +--rw policy-id?   leafref
        +--rw sites
           +--rw site* [site-id]
              +--rw site-id       svc-id
              +--rw device* [name]
              |  +--rw name    string
              |  +--rw type?   identityref
              +--rw lan-access* [name]
              |  +--rw name             string
              |  +--rw l2-technology
              |  |  +--rw l2-type?              identityref
              |  |  +--rw untagged-interface
              |  |  |  +--rw speed?   uint32
              |  |  |  +--rw mode?    neg-mode
              |  |  +--rw tagged-interface
              |  |  |  +--rw type?                identityref
              |  |  |  +--rw dot1q-vlan-tagged
              |  |  |  |  +--rw tg-type?    identityref
              |  |  |  |  +--rw cvlan-id    uint16
              |  |  |  +--rw priority-tagged
              |  |  |     +--rw tag-type?   identityref
              |  |  +--rw l2-mtu?               uint32
              |  +--rw ip-connection
              |     +--rw ipv4
              |     |  +--rw address-allocation-type?   identityref
              |     |  +--rw dhcp
              |     |  |  +--rw primary-subnet
              |     |  |  |  +--rw ip-prefix?
              |     |  |  |  |       inet:ipv4-prefix
              |     |  |  |  +--rw default-router?       inet:ip-address
              |     |  |  |  +--rw provider-addresses*
              |     |  |  |  |       inet:ipv4-address
              |     |  |  |  +--rw subscriber-address?   inet:ip-address
              |     |  |  |  +--rw reserved-ip-prefix*   inet:ip-prefix
              |     |  |  +--rw secondary-subnet* [ip-prefix]
              |     |  |     +--rw ip-prefix
              |     |  |     |       inet:ipv4-prefix
              |     |  |     +--rw provider-addresses*
              |     |  |     |       inet:ipv4-address
              |     |  |     +--rw reserved-ip-prefix*
              |     |  |             inet:ipv4-prefix
              |     |  +--rw static
              |     |     +--rw primary-subnet
              |     |     |  +--rw ip-prefix?
```

```
              |     |     |  |       inet:ipv4-prefix
              |     |     |  +--rw default-router?       inet:ip-address
              |     |     |  +--rw provider-addresses*
              |     |     |  |       inet:ipv4-address
              |     |     |  +--rw subscriber-address?   inet:ip-address
              |     |     |  +--rw reserved-ip-prefix*   inet:ip-prefix
              |     |     +--rw secondary-subnet* [ip-prefix]
              |     |        +--rw ip-prefix
              |     |        |       inet:ipv4-prefix
              |     |        +--rw provider-addresses*
              |     |        |       inet:ipv4-address
              |     |        +--rw reserved-ip-prefix*
              |     |                inet:ipv4-prefix
              |     +--rw ipv6
              |        +--rw address-allocation-type?   identityref
              |        +--rw dhcp
              |        |  +--rw subnet* [ip-prefix]
              |        |     +--rw ip-prefix
              |        |     |       inet:ipv6-prefix
              |        |     +--rw provider-addresses*
              |        |     |       inet:ipv6-address
              |        |     +--rw reserved-ip-prefix*
              |        |             inet:ipv6-prefix
              |        +--rw slaac
              |        |  +--rw subnet* [ip-prefix]
              |        |     +--rw ip-prefix
              |        |     |       inet:ipv6-prefix
              |        |     +--rw provider-addresses*
              |        |     |       inet:ipv6-address
              |        |     +--rw reserved-ip-prefix*
              |        |             inet:ipv6-prefix
              |        +--rw static
              |           +--rw subnet* [ip-prefix]
              |           |  +--rw ip-prefix
              |           |  |       inet:ipv6-prefix
              |           |  +--rw provider-addresses*
              |           |  |       inet:ipv6-address
              |           |  +--rw reserved-ip-prefix*
              |           |          inet:ipv6-prefix
              |           +--rw subscriber-address?   inet:ipv6-address
              +--rw wan-access* [name]
                 +--rw name               string
                 +--rw access-type?       identityref
                 +--rw access-provider?   string
                 +--rw bandwidth
                 |  +--rw input-bandwidth?    uint64
                 |  +--rw output-bandwidth?   uint64
                 +--rw l2-technology
```

```
                 |  +--rw l2-type?              identityref
                 |  +--rw untagged-interface
                 |  |  +--rw speed?   uint32
                 |  |  +--rw mode?    neg-mode
                 |  +--rw tagged-interface
                 |  |  +--rw type?                identityref
                 |  |  +--rw dot1q-vlan-tagged
                 |  |  |  +--rw tg-type?    identityref
                 |  |  |  +--rw cvlan-id    uint16
                 |  |  +--rw priority-tagged
                 |  |     +--rw tag-type?   identityref
                 |  +--rw l2-mtu?               uint32
                 +--rw ip-connection
                    +--rw ipv4
                    |  +--rw address-allocation-type?   identityref
                    |  +--rw dhcp
                    |  |  +--rw primary-subnet
                    |  |  |  +--rw ip-prefix?
                    |  |  |  |       inet:ipv4-prefix
                    |  |  |  +--rw default-router?       inet:ip-address
                    |  |  |  +--rw provider-addresses*
                    |  |  |  |       inet:ipv4-address
                    |  |  |  +--rw subscriber-address?   inet:ip-address
                    |  |  |  +--rw reserved-ip-prefix*   inet:ip-prefix
                    |  |  +--rw secondary-subnet* [ip-prefix]
                    |  |     +--rw ip-prefix
                    |  |     |       inet:ipv4-prefix
                    |  |     +--rw provider-addresses*
                    |  |     |       inet:ipv4-address
                    |  |     +--rw reserved-ip-prefix*
                    |  |             inet:ipv4-prefix
                    |  +--rw static
                    |     +--rw primary-subnet
                    |     |  +--rw ip-prefix?
                    |     |  |       inet:ipv4-prefix
                    |     |  +--rw default-router?       inet:ip-address
                    |     |  +--rw provider-addresses*
                    |     |  |       inet:ipv4-address
                    |     |  +--rw subscriber-address?   inet:ip-address
                    |     |  +--rw reserved-ip-prefix*   inet:ip-prefix
                    |     +--rw secondary-subnet* [ip-prefix]
                    |        +--rw ip-prefix
                    |        |       inet:ipv4-prefix
                    |        +--rw provider-addresses*
                    |        |       inet:ipv4-address
                    |        +--rw reserved-ip-prefix*
                    |                inet:ipv4-prefix
                    +--rw ipv6
```

```
                       +--rw address-allocation-type?   identityref
                       +--rw dhcp
                       |  +--rw subnet* [ip-prefix]
                       |     +--rw ip-prefix
                       |     |       inet:ipv6-prefix
                       |     +--rw provider-addresses*
                       |     |       inet:ipv6-address
                       |     +--rw reserved-ip-prefix*
                       |             inet:ipv6-prefix
                       +--rw slaac
                       |  +--rw subnet* [ip-prefix]
                       |     +--rw ip-prefix
                       |     |       inet:ipv6-prefix
                       |     +--rw provider-addresses*
                       |     |       inet:ipv6-address
                       |     +--rw reserved-ip-prefix*
                       |             inet:ipv6-prefix
                       +--rw static
                          +--rw subnet* [ip-prefix]
                          |  +--rw ip-prefix
                          |  |       inet:ipv6-prefix
                          |  +--rw provider-addresses*
                          |  |       inet:ipv6-address
                          |  +--rw reserved-ip-prefix*
                          |          inet:ipv6-prefix
                          +--rw subscriber-address?   inet:ipv6-address
```

6. YANG Modules

```
<CODE BEGINS> file "ietf-sdwan-svc@2019-06-06.yang"

module ietf-sdwan-svc {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-sdwan-svc";
  prefix sdwan-svc;

  import ietf-inet-types {
    prefix inet;
  }
  import ietf-yang-types {
    prefix yang;
  }

  organization
    "IETF foo Working Group.";
  contact
    "WG List: foo@ietf.org
     Editor:  ";
```

```
  description
    "The YANG module defines a generic service configuration
     model for Managed SD-WAN.";

  revision 2019-06-06 {
    description
      "Initial revision";
    reference "A YANG Data Model for SD-WAN service.";
  }

  typedef svc-id {
    type string;
    description
      "Type definition for service identifier";
  }

  typedef address-family {
    type enumeration {
      enum ipv4 {
        description
          "IPv4 address family.";
      }
      enum ipv6 {
        description
          "IPv6 address family.";
      }
    }
    description
      "Defines a type for the address family.";
  }

  typedef neg-mode {
    type enumeration {
      enum full-duplex {
        description
          "Defining Full duplex mode";
      }
      enum auto-neg {
        description
          "Defining Auto negotiation mode";
      }
    }
    description
      "Defining a type of the negotiation mode";
  }

  typedef device-type {
    type enumeration {
```

```
      enum physical {
        description
          "Physical device";
      }
      enum virtual {
        description
          "Virtual device";
      }
    }
    description
      "Defines device types.";
  }

  identity device-type {
    description
      "Base identity for device type.";
  }

  identity virtual-ce {
    base device-type;
    description
      "Identity for virtual-ce.";
  }

  identity physical-ce {
    base device-type;
    description
      "Identity for physical-ce.";
  }

  identity customer-application {
    description
      "Base identity for customer application.";
  }

  identity web {
    base customer-application;
    description
      "Identity for Web application (e.g., HTTP, HTTPS).";
  }

  identity mail {
    base customer-application;
    description
      "Identity for mail application.";
  }

  identity file-transfer {
```

```
    base customer-application;
    description
      "Identity for file transfer application (e.g., FTP, SFTP).";
  }

  identity database {
    base customer-application;
    description
      "Identity for database application.";
  }

  identity social {
    base customer-application;
    description
      "Identity for social-network application.";
  }

  identity games {
    base customer-application;
    description
      "Identity for gaming application.";
  }

  identity p2p {
    base customer-application;
    description
      "Identity for peer-to-peer application.";
  }

  identity network-management {
    base customer-application;
    description
      "Identity for management application
       (e.g., Telnet, syslog, SNMP).";
  }

  identity voice {
    base customer-application;
    description
      "Identity for voice application.";
  }

  identity video {
    base customer-application;
    description
      "Identity for video conference application.";
  }
```

```
  identity eth-inf-type {
    description
      "Identity of the Ethernet interface type.";
  }

  identity tagged {
    base eth-inf-type;
    description
      "Identity of the tagged interface type.";
  }

  identity untagged {
    base eth-inf-type;
    description
      "Identity of the untagged interface type.";
  }

  identity lag {
    base eth-inf-type;
    description
      "Identity of the LAG interface type.";
  }

  identity tag-type {
    description
      "Base identity from which all tag types
       are derived from";
  }

  identity c-vlan {
    base tag-type;
    description
      "A Customer-VLAN tag, normally using the 0x8100
       Ethertype";
  }

  identity tagged-inf-type {
    description
      "Identity for the tagged
       interface type.";
  }

  identity dot1q {
    base tagged-inf-type;
    description
      "Identity for dot1q vlan tagged interface.";
  }
```

```
  identity priority-tagged {
    base tagged-inf-type;
    description
      "This identity the priority-tagged interface.";
  }

  identity vpn-topology {
    description
      "Base identity for vpn topology.";
  }

  identity any-to-any {
    base vpn-topology;
    description
      "Identity for any-to-any VPN topology.";
  }

  identity hub-spoke {
    base vpn-topology;
    description
      "Identity for Hub-and-Spoke VPN topology.";
  }

  identity site-role {
    description
      "Site Role in a VPN topology ";
  }

  identity any-to-any-role {
    base site-role;
    description
      "Site in an any-to-any IP VPN.";
  }

  identity hub {
    base site-role;
    description
      "Hub Role in Hub-and-Spoke IP VPN.";
  }

  identity spoke {
    base site-role;
    description
      "Spoke Role in Hub-and-Spoke IP VPN.";
  }

  identity access-type {
    description
```

```
      "Access type of a site in a connection to different WAN";
  }

  identity commodity {
    base access-type;
    description
      "Internet access";
  }

  identity cellular {
    base access-type;
    description
      "Refers to a subset of 3G/4G/LTE and 5G";
  }

  identity private {
    base access-type;
    description
      "Refers to private circuits such as Ethernet, T1, etc";
  }

  identity routing-protocol-type {
    description
      "Base identity for routing protocol type.";
  }

  identity ospf {
    base routing-protocol-type;
    description
      "Identity for OSPF protocol type.";
  }

  identity bgp {
    base routing-protocol-type;
    description
      "Identity for BGP protocol type.";
  }

  identity static {
    base routing-protocol-type;
    description
      "Identity for static routing protocol type.";
  }

  identity address-allocation-type {
    description
      "Base identity for address-allocation-type for PE-CE link.";
  }
```

```
  identity dhcp {
    base address-allocation-type;
    description
      "Provider network provides DHCP service to customer.";
  }

  identity static-address {
    base address-allocation-type;
    description
      "Provider-to-customer addressing is static.";
  }

  identity slaac {
    base address-allocation-type;
    description
      "Use IPv6 SLAAC.";
  }

  identity ll-only {
    base address-allocation-type;
    description
      "Use IPv6 Link Local.";
  }

  identity traffic-direction {
    description
      "Base identity for traffic direction";
  }

  identity inbound {
    base traffic-direction;
    description
      "Identity for inbound";
  }

  identity outbound {
    base traffic-direction;
    description
      "Identity for outbound";
  }

  identity both {
    base traffic-direction;
    description
      "Identity for both";
  }

  identity traffic-action {
```

```
    description
      "Base identity for traffic action";
  }

  identity permit {
    base traffic-action;
    description
      "Identity for permit action";
  }

  identity deny {
    base traffic-action;
    description
      "Identity for deny action";
  }

  identity bd-limit-type {
    description
      "base identity for bd limit type";
  }

  identity percent {
    base bd-limit-type;
    description
      "Identity for percent";
  }

  identity value {
    base bd-limit-type;
    description
      "Identity for value";
  }

  identity protocol-type {
    description
      "Base identity for protocol field type.";
  }

  identity tcp {
    base protocol-type;
    description
      "TCP protocol type.";
  }

  identity udp {
    base protocol-type;
    description
      "UDP protocol type.";
```

```
  }

  identity icmp {
    base protocol-type;
    description
      "ICMP protocol type.";
  }

  identity icmp6 {
    base protocol-type;
    description
      "ICMPv6 protocol type.";
  }

  identity gre {
    base protocol-type;
    description
      "GRE protocol type.";
  }

  identity ipip {
    base protocol-type;
    description
      "IP-in-IP protocol type.";
  }

  identity hop-by-hop {
    base protocol-type;
    description
      "Hop-by-Hop IPv6 header type.";
  }

  identity routing {
    base protocol-type;
    description
      "Routing IPv6 header type.";
  }

  identity esp {
    base protocol-type;
    description
      "ESP header type.";
  }

  identity ah {
    base protocol-type;
    description
      "AH header type.";
```

```
  }

  grouping vpn-endpoint {
    leaf endpoint-id {
      type svc-id;
      description
        "Identity for the vpn endpoint";
    }
    leaf site-role {
      type identityref {
        base site-role;
      }
      default "any-to-any-role";
      description
        "Role of the site in the VPN.";
    }
    container site-attachment {
      leaf site-id {
        type leafref {
          path "/sdwan-svc/sites/site/site-id";
        }
        description
          "Defines site id attached.";
      }
      description
        "Defines site attachment to a vpn endpoint.";
    }
    container endpoint-policy-map {
      list app-group-policy {
        key "app-group-id";
        leaf app-group-id {
          type leafref {
            path "/sdwan-svc/vpn-services/vpn-service"+
            "/application-group/app-group-id";
          }
          description
            "Identity for application";
        }
        leaf policy-id {
          type leafref {
            path "/sdwan-svc/vpn-services/vpn-service/policy/policy-id";
          }
          description
            "Identity for value";
        }
        description
          "list for application group policy";
      }
```

```
      list app-policy {
        key "app-id";
        leaf app-id {
          type leafref {
            path "/sdwan-svc/vpn-services/vpn-service"+
            "/application/app-id";
          }
          description
            "Identity for application";
        }
        leaf policy-id {
          type leafref {
            path "/sdwan-svc/vpn-services/vpn-service/policy/policy-id";
          }
          description
            "Identity for value";
        }
        description
          "list for application policy";
      }
      description
        "Identity for policy maps";
    }
    description
      "grouping for vpn endpoint";
  }

  grouping flow-definition {
    container match-flow {
      leaf ethertype {
        type uint16;
        description
          "Ethertype value, e.g. 0800 for IPv4.";
      }
      leaf cvlan {
        type uint8 {
          range "0..7";
        }
        description
          "802.1Q matching.";
      }
      leaf ipv4-src-prefix {
        type inet:ipv4-prefix;
        description
          "Match on IPv4 src address.";
      }
      leaf ipv4-dst-prefix {
        type inet:ipv4-prefix;
```

```
        description
          "Match on IPv4 dst address.";
      }
      leaf l4-src-port {
        type inet:port-number;
        description
          "Match on Layer 4 src port.";
      }
      leaf l4-dst-port {
        type inet:port-number;
        description
          "Match on Layer 4 dst port.";
      }
      leaf ipv6-src-prefix {
        type inet:ipv6-prefix;
        description
          "Match on IPv6 src address.";
      }
      leaf ipv6-dst-prefix {
        type inet:ipv6-prefix;
        description
          "Match on IPv6 dst address.";
      }
      leaf protocol-field {
        type union {
          type uint8;
          type identityref {
            base protocol-type;
          }
        }
        description
          "Match on IPv4 protocol or IPv6 Next Header field.";
      }
      description
        "Describes flow-matching criteria.";
    }
    description
      "Grouping for flow definition.";
  }

  grouping application-criteria {
    list ac {
      key "name";
      ordered-by user;
      leaf name {
        type string;
        description
          "A description identifying application classification
```

```
           criteria.";
      }
      choice match-type {
        default "match-flow";
        case match-flow {
          uses flow-definition;
        }
        case match-application {
          leaf match-application {
            type identityref {
              base customer-application;
            }
            description
              "Defines the application to match.";
          }
        }
        description
          "Choice for classification.";
      }
      description
        "List of marking rules.";
    }
    description
      "This grouping defines QoS parameters for a site.";
  }

  grouping vpn-service {
    leaf vpn-id {
      type svc-id;
      description
        "Identity for VPN.";
    }
    leaf topology {
      type identityref {
        base vpn-topology;
      }
      description
        "vpn topology: hub-and-spoke or any-to-any";
    }
    container performance-objective {
      leaf start-time {
        type yang:date-and-time;
        description
          "start-time indicates date and time.";
      }
      leaf duration {
        type string;
        description
```

```
          "Time duration.";
      }
      container uptime-objective {
        leaf duration {
          type decimal64 {
            fraction-digits 5;
            range "0..100";
          }
          units "percent";
          description
            "To be used to define the a percentage of the available
             service.";
        }
        description
          "Uptime objective.";
      }
      description
        "The performance objective.";
    }
    container reserved-prefixes {
      leaf-list prefix {
        type inet:ip-prefix;
        description
          "ip prefix reserved for SP management purpose.";
      }
      description
        "ip prefix list reserved for SP management purpose.";
    }
    list application {
      key "app-id";
      leaf app-id {
        type svc-id;
        description
          "application name";
      }
      uses application-criteria;
      description
        "list for application";
    }
    list application-group {
      key "app-group-id";
      leaf app-group-id {
        type svc-id;
        description
          "application name";
      }
      leaf-list app-id {
        type leafref {
```

```
          path "../../application/app-id";
        }
        description
          "application member list in an application group";
      }
      description
        "list for application group";
    }
    list policy {
      key "policy-id";
      leaf policy-id {
        type svc-id;
        description
          "Policy names";
      }
      container policy-package {
        leaf encryption {
          type enumeration {
            enum yes {
              description
                "Indicates whether or not the application flow requires
                 to send over encrypted overlay tunnel.";
            }
            enum either {
              description
                " Either means this policy is not applied";
            }
          }
          description
            "Indicates whether or not the application flow requires
             encryption.";
        }
        leaf public-private {
          type enumeration {
            enum private-only {
              description
                "The private WAN underlay is specified.";
            }
            enum either {
              description
                "Both public WAN or private WAN could be used";
            }
          }
          description
            "Indicates whether the Application Flow can traverse
             Public or Private Underlay Connectivity Services
             (or both).Either means this policy is not applied.";
        }
```

```
        leaf local-breakout {
          type boolean;
          description
            "indicates whether the Application Flow should be
             routed directly to the Internet using Local Internet
             Breakout.It can have values Yes and No.";
        }
        leaf billing-method {
          type enumeration {
            enum flat-only {
              description
                "Only flat-rate underlay could be used for the
                 traffic.";
            }
            enum either {
              description
                "Either flat-rate or usage based underlay could
                 be used for the traffic.";
            }
          }
          description
            "billing policy.";
        }
        leaf backup-path {
          type enumeration {
            enum yes {
              description
                "Only the primary tunnel overlay could be used for
                 the traffic.";
            }
            enum no {
              description
                "Either the primary or backup overlay tunnel could be
                 used for the traffic.";
            }
          }
          description
            "overlay connection as Primary or both Primary and
             Backup.";
        }
        container bandwidth {
          leaf commit {
            type uint32;
            description
              "CIR";
          }
          leaf max {
            type uint32;
```

```
            description
              "max speed ";
          }
          description
            "Container for the bandwidth policy";
        }
        description
          "Container for policy package";
      }
      description
        "List for policy";
    }
    list endpoints {
      key "endpoint-id";
      uses vpn-endpoint;
      description
        "List of endpoints.";
    }
    description
      "Grouping of vpn service";
  }

  grouping site-l2-technology {
    container l2-technology {
      leaf l2-type {
        type identityref {
          base eth-inf-type;
        }
        default "untagged";
        description
          "Defines physical properties of an interface. By default, the
            Ethernet interface type is set to 'untagged'.";
      }
      container untagged-interface {
        leaf speed {
          type uint32;
          units "mbps";
          default "10";
          description
            "Port speed.";
        }
        leaf mode {
          type neg-mode;
          default "auto-neg";
          description
            "Negotiation mode.";
        }
        description
```

```
          "Container of Untagged Interface Attributes
           configurations.";
      }
      container tagged-interface {
        leaf type {
          type identityref {
            base tagged-inf-type;
          }
          default "dot1q";
          description
            "Tagged interface type. By default,
             the Tagged interface type is dot1q interface. ";
        }
        container dot1q-vlan-tagged {
          leaf tg-type {
            type identityref {
              base tag-type;
            }
            default "c-vlan";
            description
              "TAG type.By default, Tag type is Customer-VLAN tag.";
          }
          leaf cvlan-id {
            type uint16;
            mandatory true;
            description
              "VLAN identifier.";
          }
          description
            "Tagged interface.";
        }
        container priority-tagged {
          leaf tag-type {
            type identityref {
              base tag-type;
            }
            default "c-vlan";
            description
              "TAG type.By default, the TAG type is
               Customer-VLAN tag.";
          }
          description
            "Priority tagged.";
        }
        description
          "Container for tagged Interface.";
      }
      leaf l2-mtu {
```

```
        type uint32;
        units "bytes";
        description
          " L2 Maximum Frame Size MUST be an integer number of bytes
           >= 1522MTU.";
      }
      description
        "Container for l2 technology.";
    }
    description
      "grouping for l2 technology.";
  }

  grouping site-ip-connection {
    container ip-connection {
      container ipv4 {
        leaf address-allocation-type {
          type identityref {
            base address-allocation-type;
          }
          description
            "Defines how addresses are allocated.
             If there is no value for address
             allocation type, then the ipv4 is not enabled.";
        }
        container dhcp {
          container primary-subnet {
            leaf ip-prefix {
              type inet:ipv4-prefix;
              description
                "IPv4 address prefix and mask length between 0 and 31,
                 in bits.";
            }
            leaf default-router {
              type inet:ip-address;
              description
                "Address of default router.";
            }
            leaf-list provider-addresses {
              type inet:ipv4-address;
              description
                "the Service Provider IPv4 Addresses MUST be within the
                 specified IPv4 Prefix.";
            }
            leaf subscriber-address {
              type inet:ip-address;
              description
                "subscriber IPv4 Addresses: Non-empty list
```

```
                 of IPv4 addresses";
            }
            leaf-list reserved-ip-prefix {
              type inet:ip-prefix;
              description
                "List of IPv4 Prefixes, possibly empty";
            }
            description
              "Primary Subnet List";
          }
          list secondary-subnet {
            key "ip-prefix";
            leaf ip-prefix {
              type inet:ipv4-prefix;
              description
                "IPv4 address prefix and mask length between 0 and 31,
                 in bits";
            }
            leaf-list provider-addresses {
              type inet:ipv4-address;
              description
                "Service Provider IPv4 Addresses: Non-empty list
                 of IPv4 addresses";
            }
            leaf-list reserved-ip-prefix {
              type inet:ipv4-prefix;
              description
                "List of IPv4 Prefixes, possibly empty";
            }
            description
              "Secondary Subnet List";
          }
          description
            "DHCP allocated addresses related parameters.";
        }
        container static {
          container primary-subnet {
            leaf ip-prefix {
              type inet:ipv4-prefix;
              description
                "IPv4 address prefix and mask length between 0 and 31,
                 in bits.";
            }
            leaf default-router {
              type inet:ip-address;
              description
                "Address of default router.";
            }
```

```
            leaf-list provider-addresses {
              type inet:ipv4-address;
              description
                "the Service Provider IPv4 Addresses MUST be within the
                 specified IPv4 Prefix.";
            }
            leaf subscriber-address {
              type inet:ip-address;
              description
                "subscriber IPv4 Addresses: Non-empty list
                 of IPv4 addresses";
            }
            leaf-list reserved-ip-prefix {
              type inet:ip-prefix;
              description
                "List of IPv4 Prefixes, possibly empty";
            }
            description
              "Primary Subnet List";
          }
          list secondary-subnet {
            key "ip-prefix";
            leaf ip-prefix {
              type inet:ipv4-prefix;
              description
                "IPv4 address prefix and mask length between 0 and 31,
                 in bits";
            }
            leaf-list provider-addresses {
              type inet:ipv4-address;
              description
                "Service Provider IPv4 Addresses: Non-empty list
                 of IPv4 addresses";
            }
            leaf-list reserved-ip-prefix {
              type inet:ipv4-prefix;
              description
                "List of IPv4 Prefixes, possibly empty";
            }
            description
              "Secondary Subnet List";
          }
          description
            "Static configuration related parameters.";
        }
        description
          "IPv4-specific parameters.";
      }
```

```
      container ipv6 {
        leaf address-allocation-type {
          type identityref {
            base address-allocation-type;
          }
          description
            "Defines how addresses are allocated.
             If there is no value for address
             allocation type, then the ipv6 is not enabled.";
        }
        container dhcp {
          list subnet {
            key "ip-prefix";
            leaf ip-prefix {
              type inet:ipv6-prefix;
              description
                "IPv6 address prefix and prefix length between 0 and
                 128";
            }
            leaf-list provider-addresses {
              type inet:ipv6-address;
              description
                "Non-empty list of IPv6 addresses";
            }
            leaf-list reserved-ip-prefix {
              type inet:ipv6-prefix;
              description
                "List of IPv6 Prefixes, possibly empty";
            }
            description
              "Subnet List";
          }
          description
            "DHCP allocated addresses related parameters.";
        }
        container slaac {
          list subnet {
            key "ip-prefix";
            leaf ip-prefix {
              type inet:ipv6-prefix;
              description
                "IPv6 address prefix and prefix length of 64 ";
            }
            leaf-list provider-addresses {
              type inet:ipv6-address;
              description
                "Non-empty list of IPv6 addresses";
            }
```

```
            leaf-list reserved-ip-prefix {
              type inet:ipv6-prefix;
              description
                "List of IPv6 Prefixes, possibly empty";
            }
            description
              "Subnet List";
          }
          description
            "DHCP allocated addresses related parameters.";
        }
        container static {
          list subnet {
            key "ip-prefix";
            leaf ip-prefix {
              type inet:ipv6-prefix;
              description
                "IPv6 address prefix and prefix length between 0 and
                 128";
            }
            leaf-list provider-addresses {
              type inet:ipv6-address;
              description
                "Non-empty list of IPv6 addresses";
            }
            leaf-list reserved-ip-prefix {
              type inet:ipv6-prefix;
              description
                "List of IPv6 Prefixes, possibly empty";
            }
            description
              "Subnet List";
          }
          leaf subscriber-address {
            type inet:ipv6-address;
            description
              "IPv6 address or Not Specified.";
          }
          description
            "Static configuration related parameters.";
        }
        description
          "Describes IPv6 addresses used.";
      }
      description
        "IPv6-specific parameters.";
    }
    description
```

```
      "This grouping defines IP connection parameters.";
  }

  container sdwan-svc {
    container vpn-services {
      list vpn-service {
        key "vpn-id";
        uses vpn-service;
        description
          "List for SD-WAN";
      }
      description
        "Container for SD-WAN VPN service";
    }
    container sites {
      list site {
        key "site-id";
        leaf site-id {
          type svc-id;
          description
            "Site Name";
        }
        list device {
          key "name";
          leaf name {
            type string;
            description
              "Device Name";
          }
          leaf type {
            type identityref {
              base device-type;
            }
            description
              "Device Type: virtual or physical CE";
          }
          description
            "List for device";
        }
        list lan-access {
          key "name";
          leaf name {
            type string;
            description
              "lan access link name";
          }
          uses site-l2-technology;
          uses site-ip-connection;
```

```
          description
            "container for lan access";
        }
        list wan-access {
          key "name";
          leaf name {
            type string;
            description
              "wan access link name";
          }
          leaf access-type {
            type identityref {
              base access-type;
            }
            description
              "Access type: Internet, private VPN or cellular";
          }
          leaf access-provider {
            type string;
            description
              "Specifies the name of provider";
          }
          container bandwidth {
            leaf input-bandwidth {
              type uint64;
              description
                "input bandwidth";
            }
            leaf output-bandwidth {
              type uint64;
              description
                "output bandwidth";
            }
            description
              "Container for bandwidth";
          }
          uses site-l2-technology;
          uses site-ip-connection;
          description
            "container for wan access";
        }
        description
          "List for site";
      }
      description
        "Container for sites";
    }
    description
```

```
      "Top-level container for the SD-WAN services.";
  }
}
```

<CODE ENDS>

7. 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 NETCONF access control model [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.

8. IANA Considerations

IANA has assigned a new URI from the "IETF XML Registry" [RFC3688].

```
URI: urn:ietf:params:xml:ns:yang:ietf-sdwan-svc
Registrant Contact: The IESG
XML: N/A; the requested URI is an XML namespace.
```

IANA has recorded a YANG module name in the "YANG Module Names" registry [RFC6020] as follows:

```
Name: ietf-sdwan-svc
Namespace: urn:ietf:params:xml:ns:yang:ietf-sdwan-svc
Prefix: sdwan-svc
Reference: RFC xxxx
```

9. Appendix 1: Terminology Mapping between MEF SD-WAN Service Attributes and IETF SD-WAN model

SD-WAN Service Attributes and Services [MEF70-Draft-R1], defines the SD-WAN service attributes and services for SD-WAN service delivery. These service attributes can be used for communication between subscribers and services to deliver SD-WAN services while this draft defines a YANG data model for SD-WAN service delivery communicated between customer and service provider. The purpose of both work is very similar.

The below table shows the terminology mapping. The YANG model retains most parameter definition name but adjusts some of the structure to reserve space for future augmentation. For example, the model defines "vpn-service" and "lan-access" as a list, which can accommodate the case where the current MEF service attribute restricts only one VPN per customer and one LAN access and future extension to multiple VPN or LAN accesses per customer.

IETF SD-WAN Service model	MEF70 R1 SD-WAN Services Term
SD-WAN VPN	SD-WAN Virtual Connection (SWVC)
SD-WAN VPN Endpoint	SWVC End Point
Site	User Network Interface(UNI)
lan-access	UNI link Attributes
wan-access	TBD(Underlay connectivity)

10. Appendix 2: IETF OSE model vs IETF SD-WAN model

SD-WAN OSE service delivery model [I-D.wood-rtgwg-sdwan-ose-yang] defines two SD-WAN OSE Open SD-WAN Exchange (OSE) service YANG modules to enable the orchestrator in the enterprise network to implement SD-WAN inter-domain reachability and connectivity services and application aware traffic steering services. Although the OSE YANG model is also a service model instead of being a device model, this model is mainly used for interoperability between multiple SD-WAN domains and service consistency. The differences are shown as follows:

IETF OSE service model	IETF SD-WAN Service model
Domain SD-WAN controller facing	customer-facing
Inter OSE GW connectivity service Inter SD-WAN domain	unaware of SD-WAN domain in one SP network Inter-SD-WAN Service Provider TBD
SLA aware dynamic Path selection	static Primary/Backup selection

For the SLA based dynamic path selection policy, the OSE service model uses a similar application classification criteria, but at the same time it will collect the relevant status of the traffic SLA profiles and, based on the measurements calculated from the collected information, the primary or secondary path will be selected.

```
 +--primary-backup
      +--rw path-values
         +--rw sla-values
            +--rw latency?           uint32
            +--rw jitter?            uint32
            +--rw packet-loss-rate?  uint32
```

11. Acknowledgments

This work has benefited from the discussions of with Jack Pugaczewski, Larry S Samberg, and Pascal Menezes from MEF community.

12. Contributors

The authors would like to thank Zitao Wang for his major contributions to the initial modelling.

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>.

13.2. Informative References

[I-D.wood-rtgwg-sdwan-ose-yang]
Wood, S., Bo, W., Wu, Q., and C. Menezes, "YANG Data Model for SD-WAN OSE service delivery", draft-wood-rtgwg-sdwan-ose-yang-00 (work in progress), March 2019.

[MEF51.1] MEF, Ed., "Operator Ethernet Service Definition", December 2018, <https://wiki.mef.net/display/CESG/MEF+51.1+-+OVC+Services>.

[MEF70-Draft-R1]
MEF, Ed., "SD-WAN Service Attributes and Services", May 2019, <https://www.mef.net/Assets/Draft-Standards/MEF_70_Draft_(R1).pdf>.

[RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, DOI 10.17487/RFC2784, March 2000, <https://www.rfc-editor.org/info/rfc2784>.

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

[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>.

[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>.

[RFC4664] Andersson, L., Ed. and E. Rosen, Ed., "Framework for Layer 2 Virtual Private Networks (L2VPNs)", RFC 4664, DOI 10.17487/RFC4664, September 2006, <https://www.rfc-editor.org/info/rfc4664>.

[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>.

[RFC6071] Frankel, S. and S. Krishnan, "IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap", RFC 6071, DOI 10.17487/RFC6071, February 2011, <https://www.rfc-editor.org/info/rfc6071>.

[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>.

[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>.

[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>.

[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>.

[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>.

[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>.

Authors' Addresses

Qiong Sun
China Telecom
Beijing
China

Email: sunqiong.bri@chinatelecom.cn

Honglei Xu
China Telecom
Beijing
China

Email: xuhl.bri@chinatelecom.cn

Bo Wu (editor)
Huawei
Nanjing
China

Email: lana.wubo@huawei.com

Qin Wu (editor)
Huawei
Nanjing
China

Email: bill.wu@huawei.com

Charles Eckel (editor)
Cisco Systems
170 W. Tasman Drive
San Jose, CA
United States

Email: eckelcu@cisco.com

Network Working Group G. Zheng
Internet-Draft M. Wang
Intended status: Standards Track B. Wu
Expires: January 2, 2019 Huawei
July 1, 2018

Yang data model for Terminal Access Controller Access Control System Plus
draft-zheng-netmod-tacacs-yang-01

Abstract

This document describes a data model of Terminal Access Controller Access Control System Plus (TACACS+).

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

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document describes a data model of Terminal Access Controller Access Control System Plus (TACACS+).TACACS+ provides Device Administration for routers, network access servers and other networked computing devices via one or more centralized servers. Various TACACS+ clients and servers have been widely deployed.

This document defines a YANG [RFC7950] data model for TACACS+ draft-ietf-opsawg-tacacs-10 implementation and identification of some common properties within a device containing a Network Configuration Protocol (NETCONF) server. Devices that are managed by NETCONF and perhaps other mechanisms have common properties that need to be configured and monitored in a standard way.

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

2. Conventions used in this document

The keywords "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 BCP14, [RFC2119], [RFC8174] when, and only when, they appear in all capitals, as shown here.

The following terms are defined in [RFC6241] and are used in this specification:

o client

- o configuration data

- o server

- o state data

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

- o augment

- o data model

- o data node

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

2.1. Tree Diagrams

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

3. Problem Statement

This document defines a YANG data model which allows user to configure the TACACS+ function on a network system. YANG model can be used with network management protocols such as NETCONF [RFC6241] to install, manipulate, and delete the configuration of network devices.

TACACS+ implementations in every device may vary greatly in terms of the data hierarchy and operations that they support. Therefore this draft proposes a model that can be augmented by standard extensions and vendor proprietary models.

4. Design of the Data Model

Although different vendors have different TACACS+ data model, there is a common understanding of what Terminal Access Controller Access Control System Plus (TACACS+) is. A network system usually has a TACACS+ functions which provides centralized validation of users attempting to gain access to a device or network access server.

TACACS+ services are maintained in a database on a TACACS server.

TACACS+ provides for separate and modular authentication, authorization, and accounting facilities and allows for a single

TACACS+ server to provide each service authentication, authorization, and accounting independently. Each service can be tied into its own database to take advantage of other services available on that server or on the network, depending on the capabilities of the server.

4.1. TACACS+ Modules Overview

The ietf-tacacs+ module augments the "/sys:system" path defined in the ietf-system module [RFC7317] with "tacacs" grouping defined in Section 3.2.

Under the 'tacacs' grouping, there are global-attributes container and a tacacs-templates coantainer.

The global-attributes container is used to present the 'enable' and 'service-name' configuration and the global statistics information.

The tacacs-templates container is used to describe the tacacs configuration templates and operation templates.

Under tacacs-templates container, there are tacacs-servers container, ipv6-servers container, and host-servers container.

In the direction orthogonal to the tacacs container, presented are the commands. Those, in YANG terms, are the RPC commands. These RPC commands provide uniform APIs for resetting all statistics, resetting authentication statistics, resetting authorization statistics, resetting accounting statistics, and resetting common statistics.

The data model for tacacs has the following structure:

```
module: ietf-tacacs
  augment /sys:system:
    +--rw tacacs {tacacs}?
       +--rw global-attributes
       |  +--rw enable?            boolean
       |  +--ro total-templates?   uint32
       |  +--ro total-servers?     uint32
       |  +--rw service-name?      string
       +--rw tacacs-templates
          +--rw tacacs-template* [name]
             +--rw name                              string
             +--rw domain-include?                   boolean
             +--rw timeout?                          uint32
             +--rw quiet-time?                       uint32
             +--rw shared-key?                       password-extend
             +--rw source-ip?                        inet:ipv4-address-no-zone
             +--rw domain-mode?                      domain-include
```

```
             +--ro pri-authen-srv?               inet:ipv4-address-no-zone
             +--ro pri-common-srv?               inet:ipv4-address-no-zone
             +--ro pri-author-srv?               inet:ipv4-address-no-zone
             +--ro cur-authen-srv?               inet:ipv4-address-no-zone
             +--ro cur-author-srv?               inet:ipv4-address-no-zone
             +--ro sec-authen-srv-num?           uint32
             +--ro sec-common-srv-num?           uint32
             +--ro sec-author-srv-num?           uint32
             +--ro pri-authen-port?              uint32
             +--ro pri-common-port?              uint32
             +--ro pri-author-port?              uint32
             +--ro cur-authen-port?              uint32
             +--ro cur-author-port?              uint32
             +--ro authen-srv-connected-num?     uint32
             +--ro authen-srv-disconnected-num?  uint32
             +--ro authen-reqs-num?              uint32
             +--ro authen-rsps-num?              uint32
             +--ro authen-unknowns-num?          uint32
             +--ro authen-timeouts-num?          uint32
             +--ro authen-pkts-drop-num?         uint32
             +--ro authen-passwords-change-num?  uint32
             +--ro authen-logins-num?            uint32
             +--ro authen-send-reqs-num?         uint32
             +--ro authen-send-passwords-num?    uint32
             +--ro authen-abort-reqs-num?        uint32
             +--ro authen-connection-reqs-num?   uint32
             +--ro authen-rsp-errs-num?          uint32
             +--ro authen-rsp-fails-num?         uint32
             +--ro authen-rsp-follows-num?       uint32
             +--ro authen-get-data-num?          uint32
             +--ro authen-get-password-num?      uint32
             +--ro authen-get-user-num?          uint32
             +--ro authen-rsps-pass-num?         uint32
             +--ro authen-restart-num?           uint32
             +--ro authen-no-process-num?        uint32
             +--ro authen-time?                  uint32
             +--ro authen-errors-num?            uint32
             +--ro author-srv-connected-num?     uint32
             +--ro author-srv-disconnected-num?  uint32
             +--ro author-reqs-num?              uint32
             +--ro author-rsps-num?              uint32
             +--ro author-unknowns-num?          uint32
             +--ro author-timeouts-num?          uint32
             +--ro author-pkts-drop-num?         uint32
             +--ro author-reqs-exec-num?         uint32
             +--ro author-ppp-num?               uint32
             +--ro author-vpdn-num?              uint32
             +--ro author-rsps-err-num?          uint32
```

```
             +--ro author-rsps-exec-num?           uint32
             +--ro author-rsps-ppp-num?            uint32
             +--ro author-rsps-vpdn-num?           uint32
             +--ro author-time?                    uint32
             +--ro author-reqs-not-process-num?    uint32
             +--ro author-errors-num?              uint32
             +--ro sec-accounting-servers-num?     uint32
             +--ro cur-account-port?               uint32
             +--ro pri-account-port?               uint32
             +--ro cur-account-srv?                inet:ipv4-address-no-zone
             +--ro pri-account-srv?                inet:ipv4-address-no-zone
             +--ro account-pkts-stop-num?          uint32
             +--ro account-rsps-pass-num?          uint32
             +--ro account-rsps-num?               uint32
             +--ro account-srvs-connected-num?     uint32
             +--ro account-pkts-rsps-num?          uint32
             +--ro account-reqs-num?               uint32
             +--ro account-srv-disconnected-num?   uint32
             +--ro account-rsps-errs-num?          uint32
             +--ro account-follow-rsps-num?        uint32
             +--ro account-reqs-not-process-num?   uint32
             +--rw tacacs-servers
             |  +--rw tacacs-server* [server-ip server-type secondary-server net
work-instance public-net]
             |     +--rw server-ip                      inet:ipv4-address-no-zon
e
             |     +--rw server-type                    server-type
             |     +--rw secondary-server               boolean
             |     +--rw network-instance               -> /ni:network-instances
/network-instance/name
             |     +--rw public-net                     boolean
             |     +--rw server-port?                   uint32
             |     +--rw mux-mode-enable?               boolean
             |     +--ro server-current-state?          server-state
             |     +--ro current-srv?                   boolean
             |     +--rw shared-key?                    password-extend
             |     +--ro authen-srv-connected-num?      uint32
             |     +--ro authen-srv-disconnected-num?   uint32
             |     +--ro authen-reqs-num?               uint32
             |     +--ro authen-rsps-num?               uint32
             |     +--ro author-srv-connected-num?      uint32
             |     +--ro author-srv-disconnected-num?   uint32
             |     +--ro author-reqs-num?               uint32
             |     +--ro author-rsps-num?               uint32
             |     +--ro acct-reqs-num?                 uint32
             |     +--ro acct-rsps-num?                 uint32
             |     +--ro acct-srv-connected-num?        uint32
             |     +--ro acct-srv-disconnected-num?     uint32
             +--rw ipv6-servers
             |  +--rw ipv6-server* [server-ip server-type secondary-server netwo
rk-instance]
             |     +--rw server-ip                      inet:ipv6-address-no-zon
e
```

```
             |     +--rw server-type                    server-type
             |     +--rw secondary-server               boolean
             |     +--rw network-instance               -> /ni:network-instances
/network-instance/name
             |     +--rw server-port?                   uint32
             |     +--rw mux-mode-enable?               boolean
             |     +--ro server-state?                  server-state
             |     +--ro current-srv?                   boolean
             |     +--rw shared-key?                    password-extend
             |     +--ro authen-srv-connected-num?      uint32
             |     +--ro authen-srv-disconnected-num?   uint32
             |     +--ro authen-reqs-num?               uint32
             |     +--ro authen-rsps-num?               uint32
             |     +--ro author-srv-connected-num?      uint32
             |     +--ro author-srv-disconnected-num?   uint32
             |     +--ro author-reqs-num?               uint32
             |     +--ro author-rsps-num?               uint32
             |     +--ro acct-reqs-num?                 uint32
             |     +--ro acct-rsps-num?                 uint32
             |     +--ro acct-srv-connected-num?        uint32
             |     +--ro acct-srv-disconnected-num?     uint32
             +--rw host-servers
                +--rw host-server* [server-host-name server-type secondary-serve
r network-instance public-net]
                   +--rw server-host-name               string
                   +--rw server-type                    server-type
                   +--rw secondary-server               boolean
                   +--rw network-instance               -> /ni:network-instances
/network-instance/name
                   +--rw public-net                     boolean
                   +--rw server-port?                   uint32
                   +--rw mux-mode-enable?               boolean
                   +--ro server-state?                  server-state
                   +--ro current-server?                boolean
                   +--rw shared-key?                    password-extend
                   +--ro authen-srv-connected-num?      uint32
                   +--ro authen-srv-disconnected-num?   uint32
                   +--ro authen-reqs-num?               uint32
                   +--ro authen-rsps-num?               uint32
                   +--ro author-srv-connected-num?      uint32
                   +--ro author-srv-disconnected-num?   uint32
                   +--ro author-reqs-num?               uint32
                   +--ro author-rsps-num?               uint32
                   +--ro acct-reqs-num?                 uint32
                   +--ro acct-rsps-num?                 uint32
                   +--ro acct-srv-connected-num?        uint32
                   +--ro acct-srv-disconnected-num?     uint32

  rpcs:
    +---x rest-all-statistics
    +---x reset-authen-statistics
```

```
    +---x reset-author-statistics
    +---x reset-account-statistics
    +---x reset-common-statistics
```

5. TACACS+ Module

```
   <CODE BEGINS> file "ietf-tacacs@2018-06-25.yang"

module ietf-tacacs {
  namespace "urn:ietf:params:xml:ns:yang:ietf-tacacs";
  prefix tcs;

  import ietf-inet-types {
    prefix inet;
  }
  import ietf-network-instance {
    prefix ni;
  }
  import ietf-system {
    prefix sys;
  }

  organization
    "IETF NETMOD (NETCONF Data Modeling Language) Working Group";
  contact
    "WG Web:   <http://tools.ietf.org/wg/netmod/>
     WG List:  <mailto:netmod@ietf.org>

     Editor:   Guangying Zheng
               <mailto:zhengguangying@huawei.com>";
  description
    "This module provide defines a component that describe the
      configuration of TACACS+.";

  revision 2018-06-25 {
    description
      "Initial revision.";
    reference "foo";
  }

  typedef password-extend {
    type string {
      length "1..255";
    }
    description
      "now password extend is like string";
  }
```

```
  typedef timezone-name {
    type string;
    description
      "A time zone name as used by the Time Zone Database,
       sometimes referred to as the 'Olson Database'.

       The exact set of valid values is an implementation-specific
       matter.  Client discovery of the exact set of time zone names
       for a particular server is out of scope.";
    reference "RFC 6557: Procedures for Maintaining the Time Zone Database";
  }
  typedef server-state {
    type enumeration {
      enum "up" {
        description
          "The server is active.";
      }
      enum "down" {
        description
          "The server is inactive.";
      }
    }
    description
      "The type of tacacs server state";
  }
  typedef server-type {
    type enumeration {
      enum "authentication" {
        description
          "The server is an authentication server.";
      }
      enum "authorization" {
        description
          "The server is an authorization server.";
      }
      enum "accounting" {
        description
          "The server is an accounting server.";
      }
      enum "common" {
        description
          "The server is a common server.";
      }
    }
    description
      "The type of tacacs server";
  }
  typedef domain-include {
```

```
    type enumeration {
      enum "no" {
        description
          "User name excludes domain.";
      }
      enum "yes" {
        description
          "User name includes domain.";
      }
      enum "original" {
        description
          "User name same as user input.";
      }
    }
    description
      "The type of domain mode";
  }

  feature tacacs {
    description
      "Indicates that the device can be configured as a tacacs
       client.";
  }

  grouping tacacs {

  container tacacs {
    if-feature tacacs;
    description
      "Container for TACACS configurations and operations.";
    container global-attributes {
      description
        "TACACS global attributes.";
      leaf enable {
        type boolean;
        default "false";
        description
          "Whether the TACACS server is enabled.";
      }
      leaf total-templates {
        type uint32;
        config false;
        description
          "Total number of TACACS templates configured.";
      }
      leaf total-servers {
        type uint32;
        config false;
```

```
        description
          "Total number of TACACS servers configured.";
      }
      leaf service-name {
        type string {
          length "1..32";
        }
        description
          "TACACS service name.";
      }
    }
    container tacacs-templates {
      description
        "A set of TACACS templates.";
      list tacacs-template {
        key "name";
        description
          "List for tacacs template.";
        leaf name {
          type string;
          description
            "Name of a TACACS template, it is not case sensitive. The template n
ame can have alphabets a to z (case insensitive) and numbers from 0 to 9 or symb
ols ('.', '-' and '_').";
        }
        leaf domain-include {
          type boolean;
          default "true";
          description
            "Whether a domain name is included in a user name. By default, a use
r name contains the domain name.";
        }
        leaf timeout {
          type uint32 {
            range "1..300";
          }
          default "5";
          description
            "Server response timeout period. The default timeout period is 5 sec
onds.";
        }
        leaf quiet-time {
          type uint32 {
            range "1..255";
          }
          default "5";
          description
            "Time period after which the primary server restores to active. The
default time period is 5 minutes. The time period can be modified no matter whet
her users are using the TACACS template.";
        }
        leaf shared-key {
          type password-extend;
          description
```

```
            "Shared key for a TACACS server. Configuring a shared key improves t
he communication security between a router and TACACS server. By default, no sha
red key is configured.";
        }
        leaf source-ip {
          type inet:ipv4-address-no-zone;
          description
            "Source IP address for a TACACS server.";
        }
        leaf domain-mode {
          type domain-include;
          default "yes";
          description
            "To configure domain Mode";
        }
        leaf pri-authen-srv {
          type inet:ipv4-address-no-zone;
          config false;
          description
            "IP address of the primary authentication server.";
        }
        leaf pri-common-srv {
          type inet:ipv4-address-no-zone;
          config false;
          description
            "IP address of the primary common server.";
        }
        leaf pri-author-srv {
          type inet:ipv4-address-no-zone;
          config false;
          description
            "IP address of the primary authorization server.";
        }
        leaf cur-authen-srv {
          type inet:ipv4-address-no-zone;
          config false;
          description
            "IP address of the authentication server being used.";
        }
        leaf cur-author-srv {
          type inet:ipv4-address-no-zone;
          config false;
          description
            "IP address of authorization server being used.";
        }
        leaf sec-authen-srv-num {
          type uint32;
          config false;
          description
            "Total number of configured secondary authentication servers in the
template.";
```

```
        }
        leaf sec-common-srv-num {
          type uint32;
          config false;
          description
            "Total number of configured secondary common servers in the template
.";
        }
        leaf sec-author-srv-num {
          type uint32;
          config false;
          description
            "Total number of configured secondary authorization servers in the t
emplate.";
        }
        leaf pri-authen-port {
          type uint32;
          config false;
          description
            "Port of the primary authentication server.";
        }
        leaf pri-common-port {
          type uint32;
          config false;
          description
            "Port of the primary common server.";
        }
        leaf pri-author-port {
          type uint32;
          config false;
          description
            "Port of the primary authorization server.";
        }
        leaf cur-authen-port {
          type uint32;
          config false;
          description
            "Authentication server port being used.";
        }
        leaf cur-author-port {
          type uint32;
          config false;
          description
            "Authorization server port being used.";
        }
        leaf authen-srv-connected-num {
          type uint32;
          config false;
          description
            "Number of times that the TACACS client connected to the authenticat
ion server.";
```

```
        }
        leaf authen-srv-disconnected-num {
          type uint32;
          config false;
          description
            "Number of times that the TACACS client disconnected from the authen
tication server.";
        }
        leaf authen-reqs-num {
          type uint32;
          config false;
          description
            "Number of authentication requests. ";
        }
        leaf authen-rsps-num {
          type uint32;
          config false;
          description
            "Number of authentication responses.";
        }
        leaf authen-unknowns-num {
          type uint32;
          config false;
          description
            "Number of unknown authentication packets received by the TACACS cli
ent.";
        }
        leaf authen-timeouts-num {
          type uint32;
          config false;
          description
            "Number of times that authentication times out.";
        }
        leaf authen-pkts-drop-num {
          type uint32;
          config false;
          description
            "Number of times that authentication packets are dropped.";
        }
        leaf authen-passwords-change-num {
          type uint32;
          config false;
          description
            "Number of times that the password is changed for authentication.";
        }
        leaf authen-logins-num {
          type uint32;
          config false;
          description
            "Number of authentication logins.";
```

```
        }
        leaf authen-send-reqs-num {
          type uint32;
          config false;
          description
            "Number of authentication requests sent to server.";
        }
        leaf authen-send-passwords-num {
          type uint32;
          config false;
          description
            "Number of authentication password requests sent to the server.";
        }
        leaf authen-abort-reqs-num {
          type uint32;
          config false;
          description
            "Number of authentication abort requests sent to server.";
        }
        leaf authen-connection-reqs-num {
          type uint32;
          config false;
          description
            "Number of authentication connection requests sent to server.";
        }
        leaf authen-rsp-errs-num {
          type uint32;
          config false;
          description
            "Number of authentication error responses received from server.";
        }
        leaf authen-rsp-fails-num {
          type uint32;
          config false;
          description
            "Number of authentication response failures received from server.";
        }
        leaf authen-rsp-follows-num {
          type uint32;
          config false;
          description
            "Number of authentication Follow responses received from server.";
        }
        leaf authen-get-data-num {
          type uint32;
          config false;
          description
            "Number of authentication date responses received from server.";
```

```
        }
        leaf authen-get-password-num {
          type uint32;
          config false;
          description
            "Number of authentication password responses received from server.";
        }
        leaf authen-get-user-num {
          type uint32;
          config false;
          description
            "Number of authentication user responses received from server.";
        }
        leaf authen-rsps-pass-num {
          type uint32;
          config false;
          description
            "Number of authentication-pass responses received from server.";
        }
        leaf authen-restart-num {
          type uint32;
          config false;
          description
            "Number of authentication-restart responses received from server.";
        }
        leaf authen-no-process-num {
          type uint32;
          config false;
          description
            "Number of authentication requests that are not processed.";
        }
        leaf authen-time {
          type uint32;
          config false;
          description
            "Time (in tick) taken to complete the authentication.";
        }
        leaf authen-errors-num {
          type uint32;
          config false;
          description
            "Number of authentication errors.";
        }
        leaf author-srv-connected-num {
          type uint32;
          config false;
          description
            "Number of times that the TACACS client connected to the authorizati
on server.";
```

```
        }
        leaf author-srv-disconnected-num{
          type uint32;
          config false;
          description
            "Number of times that the TACACS client disconnected from the author
ization server.";
        }
        leaf author-reqs-num {
          type uint32;
          config false;
          description
            "Number of authorization requests. ";
        }
        leaf author-rsps-num {
          type uint32;
          config false;
          description
            "Number of authorization responses.";
        }
        leaf author-unknowns-num {
          type uint32;
          config false;
          description
            "Number of unknown authorization packets received by TACACS client."
;
        }
        leaf author-timeouts-num {
          type uint32;
          config false;
          description
            "Number of times that authorization times out.";
        }
        leaf author-pkts-drop-num {
          type uint32;
          config false;
          description
            "Number of times that authorization packets are dropped.";
        }
        leaf author-reqs-exec-num {
          type uint32;
          config false;
          description
            "Number of authorization requests for execute.";
        }
        leaf author-ppp-num {
          type uint32;
          config false;
          description
            "Number of authorization requests for PPP.";
```

```
        }
        leaf author-vpdn-num{
          type uint32;
          config false;
          description
            "Number of authorization requests for VPDN.";
        }
        leaf author-rsps-err-num {
          type uint32;
          config false;
          description
            "Number of authorization error responses.";
        }
        leaf author-rsps-exec-num {
          type uint32;
          config false;
          description
            "Number of authorization execute responses.";
        }
        leaf author-rsps-ppp-num {
          type uint32;
          config false;
          description
            "Number of authorization PPP responses.";
        }
        leaf author-rsps-vpdn-num {
          type uint32;
          config false;
          description
            "Number of authorization VPDN responses.";
        }
        leaf author-time {
          type uint32;
          config false;
          description
            "Time (in tick) taken to complete authorization.";
        }
        leaf author-reqs-not-process-num {
          type uint32;
          config false;
          description
            "Number of authorization requests that are not processed.";
        }
        leaf author-errors-num {
          type uint32;
          config false;
          description
            "Number of authorization errors.";
```

```
        }
        leaf sec-accounting-servers-num {
          type uint32;
          config false;
          description
            "Number of secondary accounting servers in the template.";
        }
        leaf cur-account-port {
          type uint32;
          config false;
          description
            "Accounting server port being used.";
        }
        leaf pri-account-port {
          type uint32;
          config false;
          description
            "Port of the primary accounting server.";
        }
        leaf cur-account-srv {
          type inet:ipv4-address-no-zone;
          config false;
          description
            "Accounting server port being used.";
        }
        leaf pri-account-srv {
          type inet:ipv4-address-no-zone;
          config false;
          description
            "Primary accounting server.";
        }
        leaf account-pkts-stop-num {
          type uint32;
          config false;
          description
            "Number of responses to accounting-stop packets.";
        }
        leaf account-rsps-pass-num {
          type uint32;
          config false;
          description
            "Number of responses to accounting-pass packets.";
        }
        leaf account-rsps-num {
          type uint32;
          config false;
          description
            "Number of responses to accounting requests.";
```

```
        }
        leaf account-srvs-connected-num {
          type uint32;
          config false;
          description
            "Number of times that the TACACS client connected to the accounting
server.";
        }
        leaf account-pkts-rsps-num {
          type uint32;
          config false;
          description
            "Number of responses to accounting-start packets.";
        }
        leaf account-reqs-num {
          type uint32;
          config false;
          description
            "Number of accounting requests sent to the server.";
        }
        leaf account-srv-disconnected-num {
          type uint32;
          config false;
          description
            "Number of times that the TACACS client disconnected from the accoun
ting server.";
        }
        leaf account-rsps-errs-num {
          type uint32;
          config false;
          description
            "Number of abnormal accounting responses received from the server.";
        }
        leaf account-follow-rsps-num {
          type uint32;
          config false;
          description
            "Number of accounting Follow responses received from server.";
        }
        leaf account-reqs-not-process-num  {
          type uint32;
          config false;
          description
            "Number of accounting requests that are not processed.";
        }
        container tacacs-servers {
          description
            "A set of TACACS servers.";
          list tacacs-server {
            key "server-ip server-type secondary-server network-instance public-
net";
```

```
            description
              "TACACS IPV4 server. A maximum 32 servers can be configured in one
 template ";

                        leaf server-ip {
              type inet:ipv4-address-no-zone;
              description
                "Server IPv4 address. Must be a valid unicast IP address.";
            }
            leaf server-type {
              type server-type;
              description
                "Server type: authentication/authorization/accounting/common.";
            }
            leaf secondary-server {
              type boolean;
              description
                "Whether the server is secondary. By default, a server is a seco
ndary server.";
            }
            leaf network-instance {
              type leafref {
                path "/ni:network-instances/ni:network-instance/ni:name";
                                }
              description
                "VPN instance name.";
            }
            leaf public-net {
              type boolean;
              description
                "Set the public-net.";
            }
            leaf server-port {
              type uint32 {
                range "1..65535";
              }
              default "49";
              description
                "Server port. Value range: 1-65535. The default port number is 4
9.";
            }
            leaf mux-mode-enable {
              type boolean;
              default "false";
              description
                "Whether the MUX mode is enabled for the server. By default, the
 MUX mode is disabled.";
            }
            leaf server-current-state {
              type server-state;
              config false;
              description
```

```
                "Server running status.";
            }
            leaf current-srv {
              type boolean;
              default "false";
              config false;
              description
                "Whether the server is being used.";
            }
            leaf shared-key {
              type password-extend;
              description
                "Shared key for a TACACS server. Configuring a shared key improv
es the communication security between a router and TACACS server. By default, no
 shared key is configured.";
            }
            leaf authen-srv-connected-num {
              type uint32;
              config false;
              description
                "Number of times that the TACACS client connected to the authent
ication server.";
            }
            leaf authen-srv-disconnected-num {
              type uint32;
              config false;
              description
                "Number of times that the TACACS client disconnected from the au
thentication server.";
            }
            leaf authen-reqs-num {
              type uint32;
              config false;
              description
                "Number of authentication requests. ";
            }
            leaf authen-rsps-num {
              type uint32;
              config false;
              description
                "Number of authentication responses.";
            }
            leaf author-srv-connected-num {
              type uint32;
              config false;
              description
                "Number of times that the TACACS client connected to the authori
zation server.";
            }
            leaf author-srv-disconnected-num {
              type uint32;
              config false;
              description
```

```
                "Number of times that the TACACS client disconnected from the au
thorization server.";
            }
            leaf author-reqs-num {
              type uint32;
              config false;
              description
                "Number of authorization requests. ";
            }
            leaf author-rsps-num {
              type uint32;
              config false;
              description
                "Number of authorization responses.";
            }
            leaf acct-reqs-num {
              type uint32;
              config false;
              description
                "Number of accounting requests. ";
            }
            leaf acct-rsps-num {
              type uint32;
              config false;
              description
                "Number of accounting responses.";
            }
            leaf acct-srv-connected-num {
              type uint32;
              config false;
              description
                "Number of times that the TACACS client connected to the account
ing server.";
            }
            leaf acct-srv-disconnected-num {
              type uint32;
              config false;
              description
                "Number of times that the TACACS client disconnected from the ac
counting server.";
            }
          }
        }
        container ipv6-servers {
          description
            "A set of TACACS servers.";
          list ipv6-server {
            key "server-ip server-type secondary-server network-instance";
            description
              "TACACS IPV6 server. A maximum 32 servers can be configured in one
 template ";
            leaf server-ip {
```

```
              type inet:ipv6-address-no-zone;
              description
                "Server IPv6 address. Must be a valid unicast IP address.";
            }
            leaf server-type {
              type server-type;
              description
                "Server type: authentication/authorization/accounting/common.";
            }
            leaf secondary-server {
              type boolean;
              description
                "Whether the server is secondary. By default, a server is a seco
ndary server.";
            }
            leaf network-instance {
              type leafref {
                path "/ni:network-instances/ni:network-instance/ni:name";
                                }
              description
                "Configure the vpn-instance name.";
            }
            leaf server-port {
              type uint32 {
                range "1..65535";
              }
              default "49";
              description
                "Server port. Value range: 1-65535. The default port number is 4
9.";
            }
            leaf mux-mode-enable {
              type boolean;
              default "false";
              description
                "Whether the MUX mode is enabled for the server. By default, the
 MUX mode is disabled.";
            }
            leaf server-state {
              type server-state;
              config false;
              description
                "Server running status.";
            }
            leaf current-srv {
              type boolean;
              default "false";
              config false;
              description
                "Whether the server is being used.";
            }
```

```
            leaf shared-key {
              type password-extend;
              description
                "Shared key for a TACACS server. Configuring a shared key improv
es the communication security between a router and TACACS server. By default, no
 shared key is configured.";
            }
            leaf authen-srv-connected-num {
              type uint32;
              config false;
              description
                "Number of times that the TACACS client connected to the authent
ication server.";
            }
            leaf authen-srv-disconnected-num {
              type uint32;
              config false;
              description
                "Number of times that the TACACS client disconnected from the au
thentication server.";
            }
            leaf authen-reqs-num {
              type uint32;
              config false;
              description
                "Number of authentication requests. ";
            }
            leaf authen-rsps-num {
              type uint32;
              config false;
              description
                "Number of authentication responses.";
            }
            leaf author-srv-connected-num {
              type uint32;
              config false;
              description
                "Number of times that the TACACS client connected to the authori
zation server.";
            }
            leaf author-srv-disconnected-num {
              type uint32;
              config false;
              description
                "Number of times that the TACACS client disconnected from the au
thorization server.";
            }
            leaf author-reqs-num{
              type uint32;
              config false;
              description
                "Number of authorization requests. ";
            }
            leaf author-rsps-num {
```

```
              type uint32;
              config false;
              description
                "Number of authorization responses.";
            }
            leaf acct-reqs-num {
              type uint32;
              config false;
              description
                "Number of accounting requests. ";
            }
            leaf acct-rsps-num {
              type uint32;
              config false;
              description
                "Number of accounting responses.";
            }
            leaf acct-srv-connected-num {
              type uint32;
              config false;
              description
                "Number of times that the TACACS client connected to the account
ing server.";
            }
            leaf acct-srv-disconnected-num {
              type uint32;
              config false;
              description
                "Number of times that the TACACS client disconnected from the ac
counting server.";
            }
          }
        }
        container host-servers {
          description
            "A set of TACACS host servers.";
          list host-server {
            key "server-host-name server-type secondary-server network-instance
public-net";
            description
              "TACACS host server. A maximum 32 servers can be configured in one
 template.";
            leaf server-host-name {
              type string {
                length "1..255";
              }
              description
                "Host name of TACACS server. Host name, Can include character '.
', '-', '_' and lowercase or uppercase letters and digit, at least include one l
etter or digit.";
            }
            leaf server-type {
              type server-type;
              description
```

```
                "Server type: authentication/authorization/accounting/common.";
            }
            leaf secondary-server {
              type boolean;
              description
                "Whether the server is secondary. By default, a server is a seco
ndary server.";
            }
            leaf network-instance {
              type leafref {
                path "/ni:network-instances/ni:network-instance/ni:name";
                                    }
              description
                "VPN instance name.";
            }
            leaf public-net {
              type boolean;
              description
                "Set the public-net.";
            }
            leaf server-port {
              type uint32 {
                range "1..65535";
              }
              default "49";
              description
                "Server port. Value range: 1-65535. The default port number is 4
9.";
            }
            leaf mux-mode-enable {
              type boolean;
              default "false";
              description
                "Whether the MUX mode is enabled for the server. By default, the
 MUX mode is disabled.";
            }
            leaf server-state {
              type server-state;
              config false;
              description
                "Server running status.";
            }
            leaf current-server {
              type boolean;
              default "false";
              config false;
              description
                "Whether the server is being used.";
            }
            leaf shared-key {
              type password-extend;
```

```
              description
                "Shared key for a TACACS server. Configuring a shared key improv
es the communication security between a router and TACACS server. By default, no
 shared key is configured.";
            }
            leaf authen-srv-connected-num {
              type uint32;
              config false;
              description
                "Number of times that the TACACS client connected to the authent
ication server.";
            }
            leaf authen-srv-disconnected-num {
              type uint32;
              config false;
              description
                "Number of times that the TACACS client disconnected from the au
thentication server.";
            }
            leaf authen-reqs-num {
              type uint32;
              config false;
              description
                "Number of authentication requests. ";
            }
            leaf authen-rsps-num {
              type uint32;
              config false;
              description
                "Number of authentication responses.";
            }
            leaf author-srv-connected-num {
              type uint32;
              config false;
              description
                "Number of times that the TACACS client connected to the authori
zation server.";
            }
            leaf author-srv-disconnected-num {
              type uint32;
              config false;
              description
                "Number of times that the TACACS client disconnected from the au
thorization server.";
            }
            leaf author-reqs-num {
              type uint32;
              config false;
              description
                "Number of authorization requests. ";
            }
            leaf author-rsps-num {
              type uint32;
              config false;
```

```
              description
                "Number of authorization responses.";
            }
            leaf acct-reqs-num {
              type uint32;
              config false;
              description
                "Number of accounting requests. ";
            }
            leaf acct-rsps-num {
              type uint32;
              config false;
              description
                "Number of accounting responses.";
            }
            leaf acct-srv-connected-num {
              type uint32;
              config false;
              description
                "Number of times that the TACACS client connected to the account
ing server.";
            }
            leaf acct-srv-disconnected-num {
              type uint32;
              config false;
              description
                "Number of times that the TACACS client disconnected from the ac
counting server.";
            }
          }
        }
      }
    }
  }
  description
  "Grouping for tacacs";
 }

 augment "/sys:system" {
  uses tacacs;
  description
  "Augment the system module";
 }

  rpc rest-all-statistics {
    description
      "Reset All Statistics.";
  }
  rpc reset-authen-statistics {
    description
```

```
        "Reset authentication statistics of the TACACS server.";
    }
    rpc reset-author-statistics {
      description
        "Reset authorization statistics of the TACACS server.";
    }
    rpc reset-account-statistics {
      description
        "Reset accounting statistics of the TACACS server.";
    }
    rpc reset-common-statistics {
      description
        "Reset common statistics of the TACACS server.";
    }
  }
```

<CODE ENDS>

6. Security Considerations

The YANG module defined in this document 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 [RFC5246].

The NETCONF access control model [RFC6536] 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.

7. IANA Considerations

This document registers a URI in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registration is requested to be made:

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

This document registers a YANG module in the YANG Module Names registry [RFC7950].

Name: ietf-tacacs
Namespace: urn:ietf:params:xml:ns:yang: ietf-tacacs
Prefix: tcs
Reference: RFC XXXX

8. Normative References

[RFC1492] Finseth, C., "An Access Control Protocol, Sometimes Called TACACS", RFC 1492, DOI 10.17487/RFC1492, July 1993, <https://www.rfc-editor.org/info/rfc1492>.

[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>.

[RFC6021] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6021, DOI 10.17487/RFC6021, October 2010, <https://www.rfc-editor.org/info/rfc6021>.

[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>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <https://www.rfc-editor.org/info/rfc6536>.

[RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, DOI 10.17487/RFC7317, August 2014, <https://www.rfc-editor.org/info/rfc7317>.

[RFC792] Postel, J., "Internet Control Message Protocol", RFC 792, September 1981.

[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>.

[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>.

[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>.

Authors' Addresses

Guangying Zheng
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: zhengguangying@huawei.com

Michael Wang
Huawei Technologies, Co., Ltd
101 Software Avenue, Yuhua District
Nanjing 210012
China

Email: wangzitao@huawei.com

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

Email: lana.wubo@huawei.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 22, 2019

G. Zheng
M. Wang
B. Wu
Huawei

June 20, 2019

Yang data model for TACACS+
draft-zheng-opsawg-tacacs-yang-02

Abstract

This document defines a YANG modules that augment the System data model defined in the RFC 7317 with TACACS+ client model. The data model of Terminal Access Controller Access Control System Plus (TACACS+) client allows the configuration of TACACS+ servers for centralized Authentication, Authorization and Accounting.

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

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document defines a YANG modules that augment the System data model defined in the [RFC7317] with TACACS+ client model.

TACACS+ provides Device Administration for routers, network access servers and other networked computing devices via one or more centralized servers which is defined inthe TACACS+ Protocol. [I-D.ietf-opsawg-tacacs]

The System Management Model [RFC7317] defines two YANG features to support local or RADIUS authentication:

- o User Authentication Model: Define a list of usernames and passwords and control the order in which local or RADIUS authentication is used.

- o RADIUS Client Model: Defines a list of RADIUS server that a device used.

Since TACACS+ is also used for device management and the feature is not contained in the system model, this document defines a YANG data model that allows users to configure TACACS+ client functions on a device for centralized Authentication, Authorization and Accounting provided by TACACS+ servers.

The YANG models can be used with network management protocols such as NETCONF[RFC6241] to install, manipulate, and delete the configuration of network devices.

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

2. Conventions used in this document

The keywords "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 BCP14, [RFC2119], [RFC8174] when, and only when, they appear in all capitals, as shown here.

The following terms are defined in [RFC6241] and are used in this specification:

- client
- configuration data
- server
- state data

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].

2.1. Tree Diagrams

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

3. TACACS+ Client Model

This model is used to configure TACACS+ client on the device to support deployment scenarios with centralized authentication, authorization, and accounting servers. Authentication is used to validates a user’s name and password, authorization allows the user to access and execute commands at various command levels assigned to the user and accounting keeps track of the activity of a user who has accessed the device.

The ietf-system-tacacsplus module is intended to augment the "/sys:system" path defined in the ietf-system module with "tacacsplus" grouping. Therefore, a device can use local, Remote Authentication Dial In User Service (RADIUS), or Terminal Access Controller Access Control System Plus (TACACS+) to validate users who attempt to access the router by several mechanisms, e.g. a command line interface or a web-based user interface.

The "server" list is directly under the "tacacsplus" container, which is to hold a list of different TACACS+ server and use server-type to distinguish the three protocols. The list of servers is for redundancy purpose.

Most of the parameters in the "server" list are taken directly from the TACACS+ protocol [I-D.ietf-opsawg-tacacs], and some are derived from the wide implementation of network equipment manufacturers. For example, when there are multiple interfaces connected to the TACACS+ server, the source address of outgoing TACACS+ packets could be specified, or the source address could be specified through the interface setting. For the TACACS + server located in a private network, a VRF instance needs to be specified.

The "statistics" container under the "server list" is to record session statistics and usage information during user access which include the amount of data a user has sent and/or received during a session.

The data model for TACACS+ client has the following structure:

```
   module: ietf-system-tacacsplus
     augment /sys:system:
       +--rw tacacsplus {tacacsplus}?
          +--rw server* [name]
             +--rw name                      string
             +--rw server-type?              enumeration
             +--rw address                   inet:host
             +--rw port?                     inet:port-number
             +--rw shared-secret             string
             +--rw (source-type)?
             |  +--:(source-ip)
             |  |  +--rw source-ip?          inet:ip-address
             |  +--:(source-interface)
             |     +--rw source-interface?   if:interface-ref
             +--rw single-connection?        boolean
             +--rw timeout?                  uint16
             +--rw vrf-instance?
             |       -> /ni:network-instances/network-instance/name
             +--ro statistics
                +--ro connection-opens?      yang:counter64
                +--ro connection-closes?     yang:counter64
                +--ro connection-aborts?     yang:counter64
                +--ro connection-failures?   yang:counter64
                +--ro connection-timeouts?   yang:counter64
                +--ro messages-sent?         yang:counter64
                +--ro messages-received?     yang:counter64
                +--ro errors-received?       yang:counter64
```

4. TACACS+ Client Module

```
   <CODE BEGINS> file "ietf-system-tacacsplus@2019-06-20.yang"

 module ietf-system-tacacsplus {
   yang-version 1.1;
   namespace "urn:ietf:params:xml:ns:yang:ietf-system-tacacsplus";
   prefix sys-tcsplus;

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

```
      "RFC 8529: YANG Data Model for Network Instances";
  }
  import ietf-interfaces {
    prefix if;
    reference
      "RFC 8343: A YANG Data Model for Interface Management";
  }
  import ietf-system {
    prefix sys;
    reference "RFC 7317: A YANG Data Model for System Management";
  }
  import ietf-netconf-acm {
    prefix nacm;
    reference "RFC 8341: Network Configuration Access Control Model";
  }

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

     Editor:   Guangying Zheng
               <mailto:zhengguangying@huawei.com>";
  description
    "This module provides configuration of TACACS+ client.

     Copyright (c) 2018 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).

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

  revision 2019-06-20 {
    description
      "Initial revision.";
    reference "foo";
  }

  feature tacacsplus {
    description
```

```
       "Indicates that the device can be configured as a TACACS+
        client.";
     reference "draft-ietf-opsawg-tacacs-11: The TACACS+ Protocol";
   }

   grouping statistics {
     description
       "Grouping for TACACS+ packets statistics attributes";
     container statistics {
       config false;
       description
         "A collection of server-related statistics objects";
       leaf connection-opens {
         type yang:counter64;
         description
           "Number of new connection requests sent to the server, e.g.
            socket open";
       }
       leaf connection-closes {
         type yang:counter64;
         description
           "Number of connection close requests sent to the server, e.g.
            socket close";
       }
       leaf connection-aborts {
         type yang:counter64;
         description
           "Number of aborted connections to the server.  These do
            not include connections that are close gracefully.";
       }
       leaf connection-failures {
         type yang:counter64;
         description
           "Number of connection failures to the server";
       }
       leaf connection-timeouts {
         type yang:counter64;
         description
           "Number of connection timeouts to the server";
       }
       leaf messages-sent {
         type yang:counter64;
         description
           "Number of messages sent to the server";
       }
       leaf messages-received {
         type yang:counter64;
         description
```

```
            "Number of messages received by the server";
        }
        leaf errors-received {
          type yang:counter64;
          description
            "Number of error messages received from the server";
        }
      }
    }

    grouping tacacsplus {
      description
        "Grouping for TACACS+ attributes";
      container tacacsplus {
        if-feature "tacacsplus";
        description
          "Container for TACACS+ configurations and operations.";
        list server {
          key "name";
          ordered-by user;
          description
            "List of TACACS+ servers used by the device

               When the TACACS+ client is invoked by a calling
               application, it sends the query to the first server in
               this list.  If no response has been received within
               'timeout' seconds, the client continues with the next
               server in the list.  If no response is received from any
               server, the client continues with the first server again.
               When the client has traversed the list 'attempts' times
               without receiving any response, it gives up and returns an
               error to the calling application.";
          leaf name {
            type string;
            description
              "An arbitrary name for the TACACS+ server.";
          }
          leaf server-type {
            type enumeration {
              enum authentication {
                description
                  "The server is an authentication server.";
              }
              enum authorization {
                description
                  "The server is an authorization server.";
              }
              enum accounting {
```

```
              description
                "The server is an accounting server.";
            }
          }
          description
            "Server type: authentication/authorization/accounting.";
        }
        leaf address {
          type inet:host;
          mandatory true;
          description
            "The address of the TACACS+ server.";
        }
        leaf port {
          type inet:port-number;
          default "49";
          description
            "The port number of TACACS+ Server port.";
        }
        leaf shared-secret {
          type string;
          mandatory true;
          nacm:default-deny-all;
          description
            "The shared secret, which is known to both the
             TACACS+ client and server. TACACS+ server administrators
             SHOULD configure secret keys of minimum
             16 characters length.";
          reference "TACACS+ protocol:";
        }
        choice source-type {
          description
            "The source address type for outbound TACACS+ packets.";
          case source-ip {
            leaf source-ip {
              type inet:ip-address;
              description
                "Specifies source IP address for TACACS+ outbound
               packets.";
            }
          }
          case source-interface {
            leaf source-interface {
              type if:interface-ref;
              description
                "Specifies the interface from which the IP address is
                 derived for use as the source for the outbound TACACS+
                 packet";
```

```
            }
          }
        }
        leaf single-connection {
          type boolean;
          default "false";
          description
            "Whether the single connection mode is enabled for the
             server. By default, the single connection mode is
             disabled.";
        }
        leaf timeout {
          type uint16 {
            range "1..300";
          }
          units "seconds";
          default "5";
          description
            "The number of seconds the device will wait for a
             response from each TACACS+ server before trying with a
             different server.";
        }
        leaf vrf-instance {
          type leafref {
            path "/ni:network-instances/ni:network-instance/ni:name";
          }
          description
            "Specifies the VPN Routing and Forwarding (VRF) instance to
             use to communicate with the TACACS+ server.";
        }

        uses statistics;
      }
    }
  }

  augment "/sys:system" {
    description
      "Augment the system model with authorization and accounting
         attributes
       Augment the system model with the tacacsplus model";
    uses tacacsplus;
  }
}

  <CODE ENDS>
```

5. Security Considerations

The YANG module defined in this document 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 NETCONF access control model [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.

This document describes the use of TACACS+ for purposes of authentication, authorization and accounting, it is vulnerable to all of the threats that are present in TACACS+ applications. For a discussion of such threats, see Section 9 of the TACACS+ Protocol [I-D.ietf-opsawg-tacacs].

6. IANA Considerations

This document registers a URI in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registration is requested to be made:

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

This document registers a YANG module in the YANG Module Names registry [RFC7950].

```
Name: ietf-system-tacacsplus
       Namespace: urn:ietf:params:xml:ns:yang: ietf-tacacsplus
       Prefix: sys-tcsplus
       Reference: RFC XXXX
```

7. Acknowledgments

The authors wish to thank Alex Campbell and Ebben Aries, Alan DeKok, Joe Clarke, many others for their helpful comments.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

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

[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>.

[RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, DOI 10.17487/RFC7317, August 2014, <https://www.rfc-editor.org/info/rfc7317>.

[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>.

[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>.

[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>.

8.2. Informative References

[I-D.ietf-opsawg-tacacs]
Dahm, T., Ota, A., dcmgash@cisco.com, d., Carrel, D., and L. Grant, "The TACACS+ Protocol", draft-ietf-opsawg-tacacs-13 (work in progress), March 2019.

Authors' Addresses

Guangying Zheng
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: zhengguangying@huawei.com

Michael Wang
Huawei Technologies, Co., Ltd
101 Software Avenue, Yuhua District
Nanjing 210012
China

Email: wangzitao@huawei.com

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

Email: lana.wubo@huawei.com