Internet Engineering Task Force (IETF)
Request for Comments: 6728
Category: Standards Track
ISSN: 2070-1721

G. Muenz
TU Muenchen
B. Claise
P. Aitken
Cisco Systems, Inc.

October 2012

# Configuration Data Model for the IP Flow Information Export (IPFIX) and Packet Sampling (PSAMP) Protocols

## Abstract

This document specifies a data model for the IP Flow Information Export (IPFIX) and Packet Sampling (PSAMP) protocols. It is for configuring and monitoring Selection Processes, Caches, Exporting Processes, and Collecting Processes of IPFIX- and PSAMP-compliant Monitoring Devices using the Network Configuration Protocol (NETCONF). The data model is defined using UML (Unified Modeling Language) class diagrams and formally specified using YANG. The configuration data is encoded in Extensible Markup Language (XML).

## Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc6728.

Table of Contents

## 1. Introduction

IPFIX- and PSAMP-compliant Monitoring Devices (routers, switches, monitoring probes, Collectors, etc.) offer various configuration possibilities that allow adapting network monitoring to the goals and purposes of the application, such as accounting and charging, traffic analysis, performance monitoring, and security monitoring. The use of a common vendor-independent configuration data model for IPFIX- and PSAMP-compliant Monitoring Devices facilitates network management and configuration, especially if Monitoring Devices of different implementers or manufacturers are deployed simultaneously. On the one hand, a vendor-independent configuration data model helps to store and manage the configuration data of Monitoring Devices in a consistent format. On the other hand, it can be used for local and remote configuration of Monitoring Devices.

The purpose of this document is the specification of a vendor-independent configuration data model that covers the commonly available configuration parameters of Selection Processes, Caches, Exporting Processes, and Collecting Processes. In addition, it includes common states parameters of a Monitoring Device. The configuration data model is defined using UML (Unified Modeling Language) class diagrams [UML], while the actual configuration data is encoded in Extensible Markup Language (XML) [W3C.REC-xml-20081126]. An XML document conforming to the configuration data model contains the configuration data of one Monitoring Device.

The configuration data model is designed for use with the NETCONF protocol [RFC6241] in order to configure remote Monitoring Devices. With the NETCONF protocol, it is possible to transfer a complete set of configuration data to a Monitoring Device, to query the current configuration and state parameters of a Monitoring Device, and to change specific parameter values of an existing Monitoring Device configuration.

In order to ensure compatibility with the NETCONF protocol [RFC6241], YANG [RFC6020] is used to formally specify the configuration data model. If required, the YANG specification of the configuration data model can be converted into XML Schema language [W3C.REC-xmlschema-0-20041028] or DSDL (Document Schema Definition Languages) [RFC6110], for example, by using the pyang tool [YANG-WEB]. YANG provides mechanisms to adapt the configuration data model to device-specific constraints and to augment the model with additional device-specific or vendor-specific parameters.

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].

## 1.1. IPFIX Documents Overview

The IPFIX protocol [RFC5101] provides network administrators with access to IP Flow information. The architecture for the export of measured IP Flow information out of an IPFIX Exporting Process to a Collecting Process is defined in [RFC5470], per the requirements defined in [RFC3917]. The IPFIX protocol [RFC5101] specifies how IPFIX Data Records and Templates are carried via a number of transport protocols from IPFIX Exporting Processes to IPFIX Collecting Process. IPFIX has a formal description of IPFIX Information Elements, their name, type, and additional semantic information, as specified in [RFC5102]. [RFC6615] specifies the IPFIX Management Information Base, consisting of the IPFIX MIB module and the IPFIX SELECTOR MIB module. Finally, [RFC5472] describes what type of applications can use the IPFIX protocol and how they can use the information provided. It furthermore shows how the IPFIX framework relates to other architectures and frameworks. Methods for efficient export of bidirectional Flow information and common properties in Data Records are specified in [RFC5103] and [RFC5473], respectively. [RFC5610] addresses the export of extended type information for enterprise-specific Information Elements. The storage of IPFIX Messages in a file is specified in [RFC5655].

### 1.2. PSAMP Documents Overview

The framework for packet selection and reporting [RFC5474] enables network elements to select subsets of packets by statistical and other methods, and to export a stream of reports on the selected packets to a Collector. The set of packet selection techniques (Sampling, Filtering, and hashing) standardized by PSAMP is described in [RFC5475]. The PSAMP protocol [RFC5476] specifies the export of packet information from a PSAMP Exporting Process to a PSAMP Collector. Instead of exporting PSAMP Packet Reports, the stream of selected packets may also serve as input to the generation of IPFIX Flow Records. Like IPFIX, PSAMP has a formal description of its Information Elements, their name, type, and additional semantic information. The PSAMP information model is defined in [RFC5477]. [RFC6727] specifies the PSAMP MIB module as an extension of the IPFIX SELECTOR MIB module defined in [RFC6615].

## 2. Terminology

This document adopts the terminologies used in [RFC5101], [RFC5103], [RFC5655], and [RFC5476]. As in these documents, all specific terms have the first letter of a word capitalized when used in this document. The following listing indicates in which references the definitions of those terms that are commonly used throughout this document can be found:

- Definitions adopted from [RFC5101]:
  - Collection Process
  - Collector
  - Data Record
  - Exporter
  - Flow
  - Flow Key
  - Flow Record
  - Information Element
  - IPFIX Device
  - IPFIX Message
  - Observation Domain
  - Observation Point
  - (Options) Template

- Definitions adopted from [RFC5103]:
  - Reverse Information Element

- Definitions adopted from [RFC5655]:
  - File Reader
  - File Writer

- Definitions adopted from [RFC5476]:
  - Filtering
  - Observed Packet Stream
  - Packet Report
  - PSAMP Device
  - Sampling
  - Selection Process
  - Selection Sequence
  - Selection Sequence Report Interpretation
  - Selection Sequence Statistics Report Interpretation
  - Selection State
  - Selector, Primitive Selector, Composite Selector
  - Selector Report Interpretation

The terms Metering Process and Exporting Process have different definitions in [RFC5101] and [RFC5476]. In the scope of this document, these terms are used according to the following definitions, which cover the deployment in both PSAMP Devices and IPFIX Devices:

Metering Process

> The Metering Process generates IPFIX Flow Records or PSAMP Packet Reports, depending on its deployment as part of an IPFIX Device or PSAMP Device. Inputs to the process are packets observed at one or more Observation Points, as well as characteristics describing the packet treatment at these Observation Points. If IPFIX Flow Records are generated, the Metering Process MUST NOT aggregate packets observed at different Observation Domains in the same Flow. The function of the Metering Process is split into two functional blocks: Selection Process and Cache.

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.

In addition to the existing IPFIX and PSAMP terminology, the following terms are defined:

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.

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.

Selected Packet Stream

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

## 3. Structure of the Configuration Data Model

The IPFIX reference model in [RFC5470] describes Metering Processes, Exporting Processes, and Collecting Processes as functional blocks of IPFIX Devices. The PSAMP framework [RFC5474] provides the corresponding information for PSAMP Devices and introduces the Selection Process as a functional block within Metering Processes. In Section 2 of the document, the Cache is defined as another functional block within Metering Processes. Further explanations about the relationship between Selection Process and Cache are given in Section 3.1. IPFIX File Reader and File Writer are defined as specific kinds of Exporting and Collecting Processes in [RFC5655].

Monitoring Device implementations usually maintain the separation of various functional blocks, although they do not necessarily implement all of them. Furthermore, they provide various configuration possibilities; some of them are specified as mandatory by the IPFIX

protocol [RFC5101] or PSAMP protocol [RFC5476]. The configuration data model enables the setting of commonly available configuration parameters for Selection Processes, Caches, Exporting Processes, and Collecting Processes. In addition, it allows specifying the composition of functional blocks within a Monitoring Device configuration and their linkage with Observation Points.

The selection of parameters in the configuration data model is based on configuration issues discussed in the IPFIX and PSAMP documents [RFC3917], [RFC5101], [RFC5470], [RFC5476], [RFC5474], and [RFC5475]. Furthermore, the structure and content of the IPFIX MIB module [RFC6615] and the PSAMP MIB module [RFC6727] have been taken into consideration. Consistency between the configuration data model and the IPFIX and PSAMP MIB modules is an intended goal. Therefore, parameters in the configuration data model are named according to corresponding managed objects. Certain IPFIX MIB objects containing state data have been adopted as state parameters in the configuration data model. State parameters cannot be configured, yet their values can be queried from the Monitoring Device by a network manager.

Section 3.2 explains how UML class diagrams are deployed to illustrate the structure of the configuration data model. Thereafter, Section 3.3 and Section 3.4 explain the class diagrams for the configuration of Exporters and Collectors, respectively. Each of the presented classes contains specific configuration parameters that are specified in Section 4. Section 5 gives a short introduction to YANG concepts that allow adapting the configuration data model to the capabilities of a device. The formal definition of the configuration data model in YANG is given in Section 6. Section 7 illustrates the usage of the model with example configurations in XML.

### 3.1. Metering Process Decomposition in Selection Process and Cache

In a Monitoring Device implementation, the functionality of the Metering Process is commonly split into packet Sampling and Filtering functions performed by Selection Processes, and the maintenance of Flow Records and Packet Reports is performed by a Cache. Figure 1 illustrates this separation with the example of a basic Metering Process.

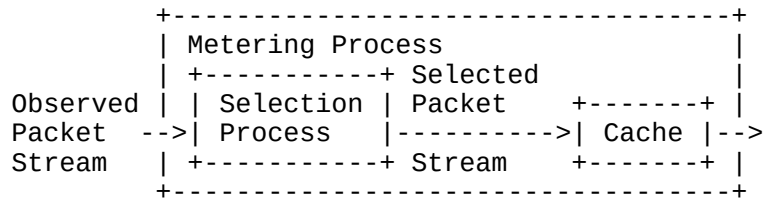


Figure 1: Selection Process and Cache forming a Metering Process

The configuration data model adopts the separation of Selection Processes and Caches in order to support the flexible configuration and combination of these functional blocks. As defined in [RFC5476], 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.

The configuration data model does not allow configuring a Metering Process without any Selection Process in front of the Cache. If all packets in the Observed Packet Stream shall be selected and passed to the Cache without any Filtering or Sampling, a Selection Process needs to be configured with a Selector that selects all packets ("SelectAll" class in Section 4.2.1).

The configuration data model enables the configuration of a Selection Process that receives packets from multiple Observation Points as its input. 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 more than one Selection Processes whose output is processed by a single Cache. This is illustrated in Figure 2.

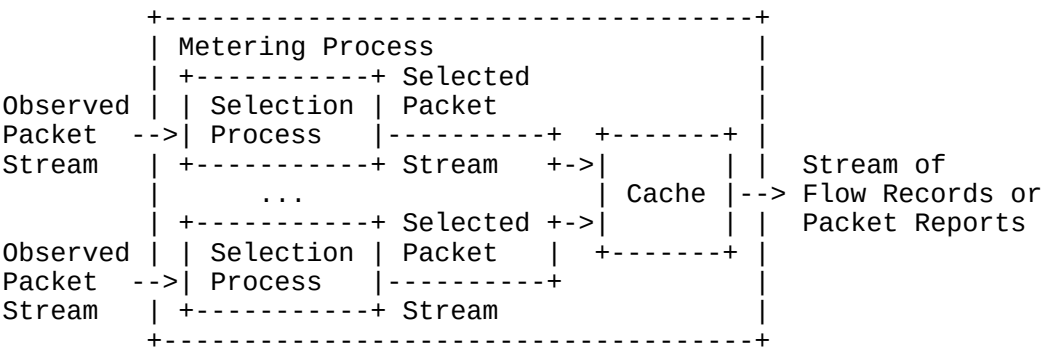


Figure 2: Metering Process with multiple Selection Processes

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 [RFC5101]), however, 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.

## 3.2. UML Representation

We use UML class diagrams [UML] to explain the structure of the configuration data model. The attributes of the classes are the configuration or state parameters. The configuration and state parameters of a given Monitoring Device are represented as objects of these classes encoded in XML.

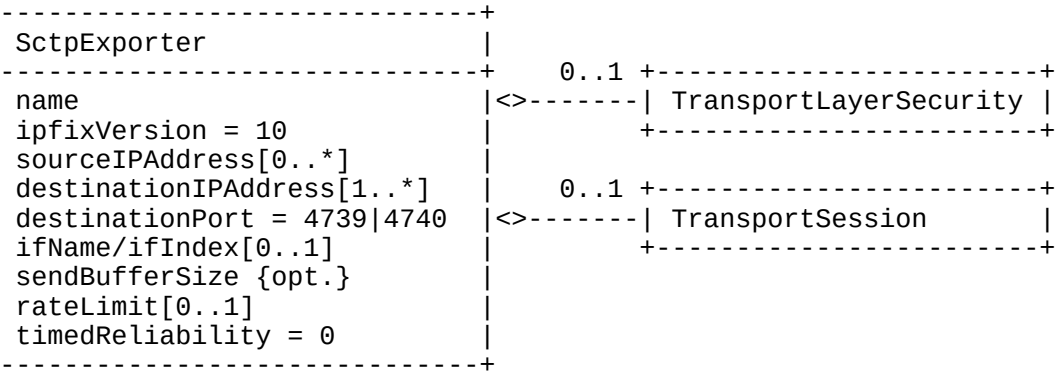


Figure 3: UML example: SctpExporter class

As an example, Figure 3 shows the UML diagram of the SctpExporter class, which is specified in more detail in Section 4.4.1. The upper box contains the name of the class. The lower box lists the attributes of the class. Each attribute corresponds to a parameter of the configuration data model.

Behind an attribute's name, there may appear a multiplicity indicator in brackets (i.e., between "[" and "]"). An attribute with multiplicity indicator "[0..1]" represents an OPTIONAL configuration parameter that is only included in the configuration data if the user configures it. Typically, the absence of an OPTIONAL parameter has a specific meaning. For example, not configuring rateLimit in an object of the SctpExporter class means that no rate limiting will be applied to the exported data. In YANG, an OPTIONAL parameter is specified as a "leaf" without "mandatory true" substatement. The "description" substatement specifies the behavior for the case that the parameter is not configured.

The multiplicity indicator "[0..*]" means that this parameter is OPTIONAL and MAY be configured multiple times with different values. In the example, multiple source IP addresses (sourceIPAddress) may be configured for a multihomed Exporting Process. In YANG, an attribute with multiplicity indicator "[0..*]" corresponds to a "leaf-list".

The multiplicity indicator "[1..*]" means that this parameter MUST be configured at least once and MAY be configured multiple times with different values. In the example, one or more destination IP addresses (destinationIPAddress) must be configured to specify the export destination. In YANG, an attribute with multiplicity indicator "[1..*]" corresponds to a "leaf-list" with "min-elements 1" substatement. Note that attributes without this multiplicity indicator MUST NOT appear more than once in each object of the class.

Attributes without multiplicity indicator may be endued with a default value that is indicated behind the equality symbol ("="). If a default value exists, the parameter does not have to be explicitly configured by the user. If the parameter is not configured by the user, the Monitoring Device MUST use the specified default value for the given parameter. In the example, IPFIX version 10 must be used unless a different value is configured for ipfixVersion. In YANG, an attribute with default value corresponds to a "leaf" with "default" substatement.

In the example, there exist two default values for the destination port (destinationPort) -- namely, the registered ports for IPFIX with and without transport layer security (i.e., DTLS or TLS), which are 4740 and 4739, respectively. In the UML diagram, the two default values are separated by a vertical bar ("|"). In YANG, such

conditional default value alternatives cannot be specified formally. Instead, they are defined in the "description" substatement of the "leaf".

Further attribute properties are denoted in braces (i.e., between "{" and "}"). An attribute with property "{opt.}", such as sendBufferSize in the SctpExporter class, represents a parameter that MAY be configured by the user. If not configured by the user, the Monitoring Device MUST set an appropriate value for this parameter at configuration time. As a result, the parameter will always exist in the configuration data, yet it is not mandatory for the user to configure it. This behavior can be implemented as a static device-specific default value, but does not have to be. Therefore, the user MUST NOT expect that the device always sets the same values for the same parameter. Regardless of whether the parameter value has been configured by the user or set by the device, the parameter value MUST NOT be changed by the device after configuration. Since this behavior cannot be specified formally in YANG, it is specified in the "description" substatement of the "leaf".

The availability of a parameter may depend on another parameter value. In the UML diagram, such restrictions are indicated as attribute properties (e.g., "{SCTP only}"). The given example does not show such restrictions. In YANG, the availability of a parameter is formally restricted with the "when" substatement of the "leaf".

Another attribute property not shown in the example is "{readOnly}", which specifies state parameters that cannot be configured. In YANG, this corresponds to the "config false" substatement.

Attributes without multiplicity indicator, without default value, and without "{readOnly}" property are mandatory configuration parameters. These parameters MUST be configured by the user unless an attribute property determines that the parameter is not available. In YANG, a mandatory parameter corresponds to a "leaf" with "mandatory true" substatement. In the example, the user MUST configure the name parameter.

If some parameters are related to each other, it makes sense to group these parameters in a subclass. This is especially useful if different subclasses represent choices of different parameter sets, or if the parameters of a subclass may appear multiple times. For example, the SctpExporter class MAY contain the parameters of the TransportLayerSecurity subclass.

An object of a class is encoded as an XML element. In order to distinguish between classes and objects, class names start with an uppercase character while the associated XML elements start with

lowercase characters. Parameters appear as XML elements that are nested in the XML element of the object. In XML, the parameters of an object can appear in any order and do not have to follow the order in the UML class diagram. Unless specified differently, the order in which parameters appear does not have a meaning. As an example, an object of the SctpExporter class corresponds to one occurrence of

```
<sctpExporter>
  <name>my-sctp-export</name>
  ...
</sctpExporter>
```

There are various possibilities how objects of classes can be related to each other. In the scope of this document, we use two different types of relationship between objects: aggregation and unidirectional association. In UML class diagrams, two different arrow types are used as shown in Figure 4.

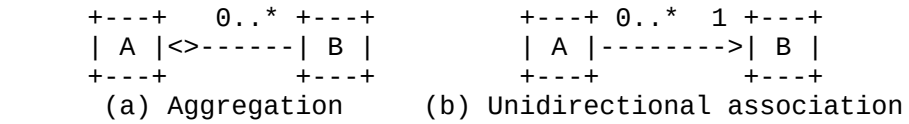


Figure 4: Class relationships in UML class diagrams

Aggregation means that one object is part of the other object. In Figure 4 (a), an object of class B is part of an object of class A. This corresponds to nested XML elements:

```
<a>
  <b>
    ...
  </b>
  ...
</a>
```

In the example, objects of the TransportLayerSecurity class and the TransportSession class appear as nested XML elements <transportLayerSecurity> and <transportSession> within an object of the SctpExporter class <sctpExporter>.

A unidirectional association is a reference to an object. In Figure 4(b), an object of class A contains a reference to an object of class B. This corresponds to separate XML elements that are not nested. To distinguish different objects of class B, class B must have a key. In the configuration data model, keys are string parameters called "name", corresponding to XML elements <name>. The names MUST be unique within the given XML subtree. The reference to

a specific object of class B is encoded with an XML element <b>, which contains the name of an object. If an object of class A refers to the object of class B with name "foo", this looks as follows:

```
<a>
  ...
  <b>foo</b>
  ...
</a>

<b>
  <name>foo</name>
  ...
</b>
```

In Figure 4, the indicated numbers define the multiplicity:

"1": one only
"0..*": zero or more
"1..*": one or more

In the case of aggregation, the multiplicity indicates how many objects of one class may be included in one object of the other class. In Figure 4(a), an object of class A may contain an arbitrary number of objects of class B. In the case of unidirectional association, the multiplicity at the arrowhead specifies the number of objects of a given class that may be referred to. The multiplicity at the arrow tail specifies how many different objects of one class may refer to a single object of the other class. In Figure 4(b), an object of class A refers to single object of class B. One object of class B can be referred to from an arbitrary number of objects of class A.

Similar to classes that are referenced in UML associations, classes that contain configuration parameters and that occur in an aggregation relationship with multiplicity greater than one must have a key. This key is necessary because every configuration parameter must be addressable in order to manipulate or delete it. The key values MUST be unique in the given XML subtree (i.e., unique within the aggregating object). Hence, if class B in Figure 4(a) contains a configuration parameter, all objects of class B belonging to the same object of class A must have different key values. Again, the key appears as an attribute called "name" in the concerned classes.

A class that contains state parameters but no configuration parameters, such as the Template class (see Section 4.8), does not have a key. This is because state parameters cannot be manipulated or deleted, and therefore do not need to be addressable.

Note that the usage of keys as described above is required by YANG [RFC6020], which mandates the existence of a key for elements that appear in a list of configuration data.

The configuration data model for IPFIX and PSAMP makes use of unidirectional associations to specify the data flow between different functional blocks. For example, if the output of a Selection Process is processed by a Cache, this corresponds to an object of the SelectionProcess class that contains a reference to an object of the Cache class. The configuration data model does not mandate that such a reference exists for every functional block that has an output. If such a reference is absent, the output is dropped without any further processing. Although such configurations are incomplete, we do not consider them invalid as they may temporarily occur if a Monitoring Device is configured in multiple steps. Also, it might be useful to pre-configure certain functions of a Monitoring Device in order to be able to switch to a new configuration more quickly.

## 3.3. Exporter Configuration

Figure 5 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:

- The ObservationPoint class specifies an Observation Point (i.e., an interface or linecard) of the Monitoring Device at which packets are captured for traffic measurements. An object of the ObservationPoint class may be associated with one or more objects of the SelectionProcess class configuring Selection Processes that process the observed packets in parallel. As long as an ObservationPoint object is specified without any references to SelectionProcess objects, the captured packets are not considered by any Metering Process.

- 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. Therefore, a SelectionProcess object MAY be referred to from one or more ObservationPoint objects.

A Selection Process MAY pass the Selected Packet Stream to a Cache. Therefore, the SelectionProcess class contains a reference to an object 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 object of the Cache class MAY be referred to from multiple SelectionProcess objects.

  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 objects of the ExportingProcess class. If a Cache object does not specify any reference to an ExportingProcess object, 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 object of the ExportingProcess class MAY be referred to from multiple objects of the Cache class.

  An Exporting Process MAY be configured as a File Writer according to [RFC5655].

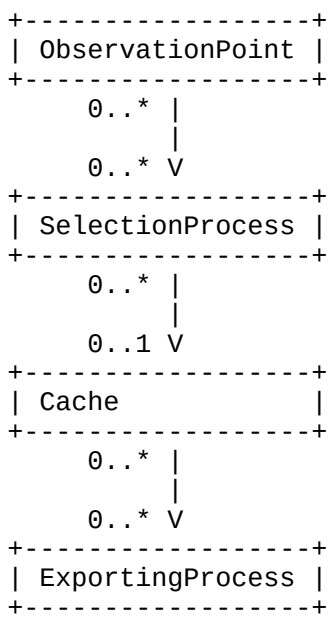


Figure 5: Class diagram of Exporter configuration

## 3.4. Collector Configuration

Figure 6 below shows the main classes of the configuration data model that are involved in the configuration of a Collector. An object of the CollectingProcess class specifies the local IP addresses, transport protocols, and port numbers of a Collecting Process. Alternatively, the Collecting Process MAY be configured as a File Reader according to [RFC5655].

An object of the CollectingProcess class may refer to one or more ExportingProcess objects configuring Exporting Processes that reexport 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.

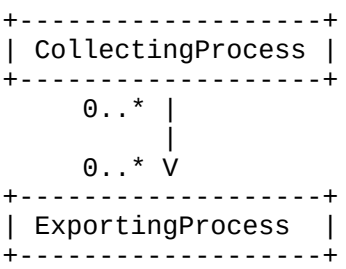


Figure 6: Class diagram of Collector configuration

## 4. Configuration Parameters

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

### 4.1. ObservationPoint Class

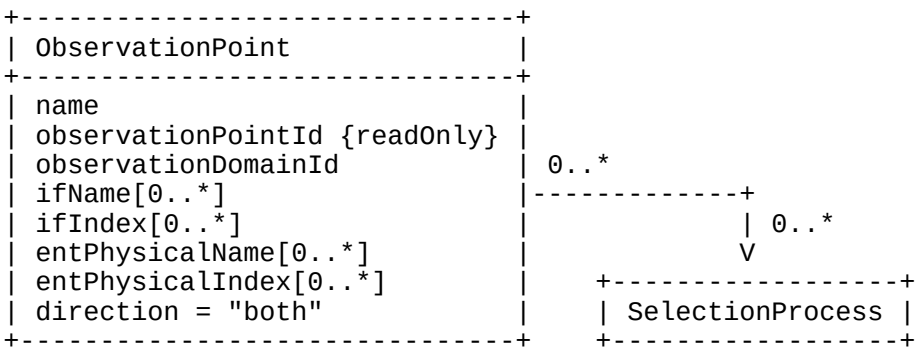


Figure 7: ObservationPoint class

Figure 7 shows the ObservationPoint class that specifies an Observation Point of the Monitoring Device.

As defined in [RFC5101], an Observation Point can be any location where packets are observed. A Monitoring Device potentially has more than one such location. An instance of ObservationPoint class defines which location is associated with a specific Observation Point. For this purpose, interfaces and physical entities 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 [RFC4133]) 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 [RFC5101], an Observation Point may be associated with a set of interfaces. Therefore, the configuration data model allows configuring multiple interfaces and physical entities 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. It appears as a state parameter in the ObservationPoint class.

The configuration parameters of the Observation Point are:

observationDomainId: This parameter defines the identifier of the Observation Domain 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].

ifName/ifIndex/entPhysicalName/entPhysicalIndex: These parameters identify interfaces and physical entities (e.g., linecards) that are on the Monitoring Device and are associated with the given Observation Point.
An interface is either identified by its name (ifName) or the ifIndex value of the corresponding object in the IF-MIB module [RFC2863]. ifIndex SHOULD only be used if an SNMP agent enables access to the ifTable.
Similarly, a physical entity is either identified by its name (entPhysicalName) or the entPhysicalIndex value of the corresponding object in the ENTITY-MIB module [RFC4133]. entPhysicalIndex SHOULD only be used if an SNMP agent enables access to the entPhysicalTable.
Note that the parameters ifIndex and entPhysicalIndex correspond to ipfixObservationPointPhysicalInterface and ipfixObservationPointPhysicalEntity in the IPFIX MIB module [RFC6615].

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.

An ObservationPoint object MAY refer to one or more SelectionProcess objects configuring Selection Processes that process the observed packets in parallel.

## 4.2. SelectionProcess Class

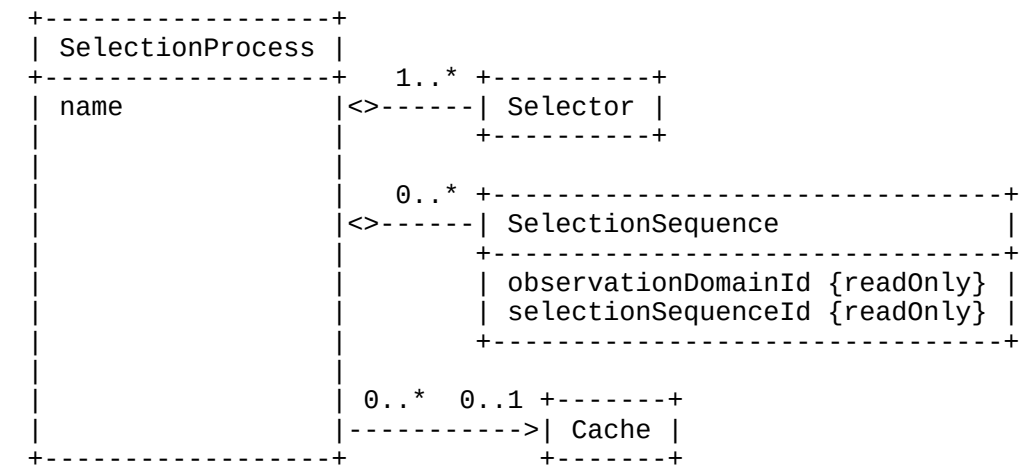


Figure 8: SelectionProcess class

Figure 8 shows the SelectionProcess class. The SelectionProcess 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 SelectionProcess class contains a list of (observationDomainId, selectionSequenceId) 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 object of the SelectionProcess class.

A SelectionProcess object MAY include a reference to an object of the Cache class to generate Packet Reports or Flow Records from the Selected Packet Stream.

### 4.2.1. Selector Class

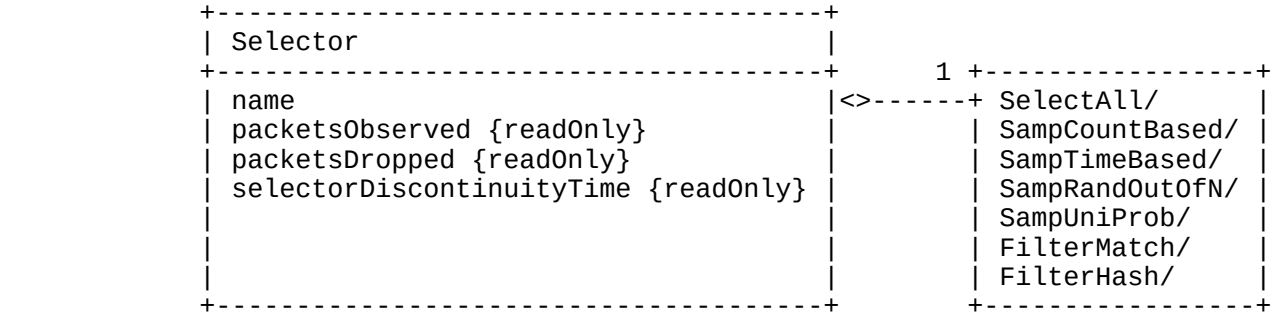


Figure 9: Selector class

The Selector class in Figure 9 contains the configuration and state parameters of a Selector. Standardized PSAMP Sampling and Filtering methods are described in [RFC5475]; their configuration parameters are specified in the classes SampCountBased, SampTimeBased, SampRandOutOfN, SampUniProb, FilterMatch, and FilterHash. In addition, the SelectAll 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.

As state parameters, the Selector class contains the Selector statistics packetsObserved and packetsDropped as well as selectorDiscontinuityTime, which correspond to the IPFIX MIB module objects ipfixSelectionProcessStatsPacketsObserved, ipfixSelectionProcessStatsPacketsDropped, and ipfixSelectionProcessStatsDiscontinuityTime, respectively [RFC6615]:

packetsObserved: 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 selectorDiscontinuityTime.

packetsDropped: 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.

selectorDiscontinuityTime: 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 sysUpTime.

Note that packetsObserved and packetsDropped 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.

#### 4.2.2. Sampler Classes

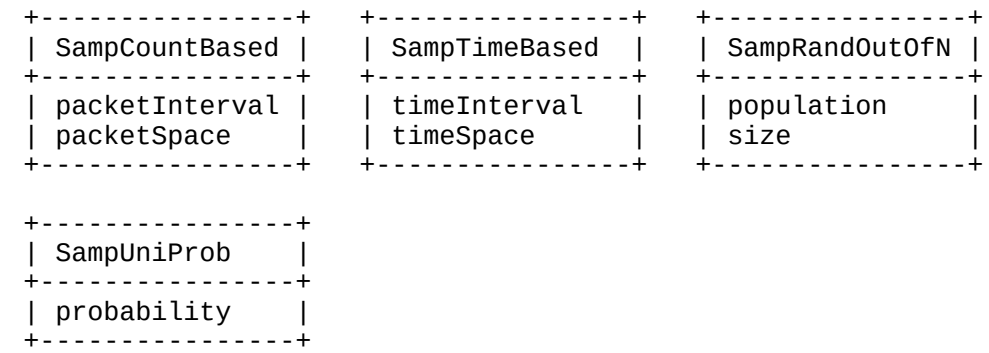


Figure 10: Sampler classes

The Sampler classes in Figure 10 contain the configuration parameters of specific Sampling algorithms:

packetInterval, packetSpace: For systematic count-based Sampling, packetInterval defines the number of packets that are consecutively sampled between gaps of length packetSpace. These parameters correspond to the Information Elements samplingPacketInterval and samplingPacketSpace [RFC5477], as well as to the PSAMP MIB objects psampSampCountBasedInterval and psampSampCountBasedSpace [RFC6727].

timeInterval, timeSpace: For systematic time-based Sampling, timeInterval defines the time interval during which all arriving packets are sampled. timeSpace is the gap between two Sampling intervals. These parameters correspond to the Information Elements samplingTimeInterval and samplingTimeSpace [RFC5477], as well as to the PSAMP MIB objects psampSampTimeBasedInterval and psampSampTimeBasedSpace [RFC6727]. The unit is microseconds.

size, population: For n-out-of-N random Sampling, size defines the number of elements taken from the parent population. population defines the number of elements in the parent population. These parameters correspond to the Information Elements samplingSize and samplingPopulation [RFC5477], as well as to the PSAMP MIB objects psampSampRandOutOfNSize and psampSampRandOutOfNPopulation [RFC6727].

probability: For uniform probabilistic Sampling, probability defines the Sampling probability. The probability is expressed as a value between 0 and 1. This parameter corresponds to the Information Element samplingProbability [RFC5477], as well as to the PSAMP MIB object psampSampUniProbProbability [RFC6727].

### 4.2.3. Filter Classes

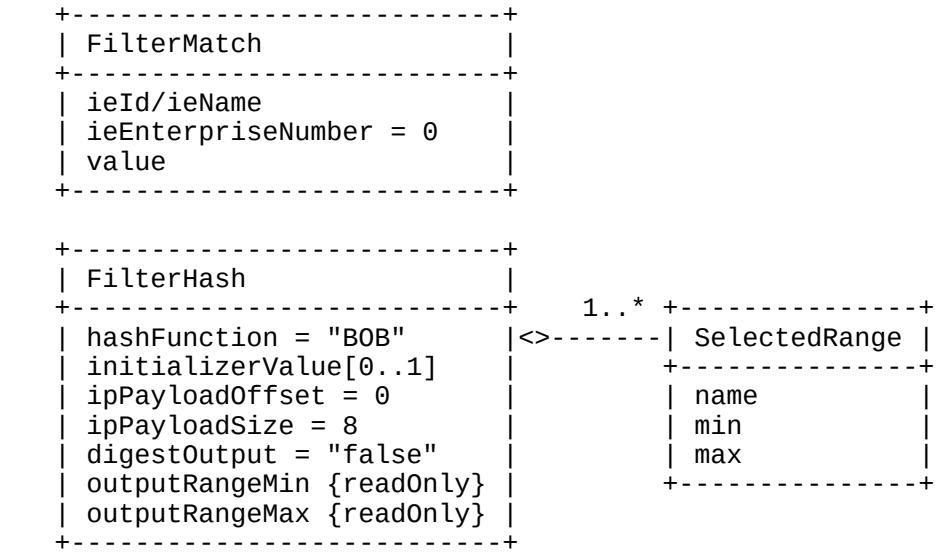


Figure 11: Filter classes

The Filter classes in Figure 11 contain the configuration parameters of specific Filtering methods. For property match Filtering, the configuration parameters are:

ieId, ieName, ieEnterpriseNumber: The property to be matched is specified by either ieId or ieName, specifying the identifier or name of the Information Element, respectively. If ieEnterpriseNumber 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 ieEnterpriseNumber specifies an enterprise-specific Information Element [IANA-ENTERPRISE-NUMBERS].

value: Matching value.

For hash-based Filtering, the configuration and state parameters are:

hashFunction: Hash function to be used. The following parameter values are defined by the configuration data model:
* 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].

initializerValue: Initializer value to the hash function. 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.

ipPayloadOffset, ipPayloadSize: ipPayloadOffset and ipPayloadSize configure the offset and the size of the payload section used as input to the hash function. Default values are 0 and 8, respectively, corresponding to the minimum configurable values according to [RFC5476], Section 6.5.2.6. These parameters correspond to the Information Elements hashIPPayloadOffset and hashIPPayloadSize [RFC5477], as well as to the PSAMP MIB objects psampFiltHashIpPayloadOffset and psampFiltHashIpPayloadSize [RFC6727].

digestOutput: digestOutput 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 digestHashValue field. This parameter corresponds to the Information Element hashDigestOutput [RFC5477].

outputRangeMin, outputRangeMax: The values of these two state parameters are the beginning and end of the hash function's potential output range. These parameters correspond to the Information Elements hashOutputRangeMin and hashOutputRangeMax [RFC5477], as well as to the PSAMP MIB objects psampFiltHashOutputRangeMin and psampFiltHashOutputRangeMax [RFC6727].

One or more ranges of matching hash values are defined by the min and max parameters of the SelectedRange 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

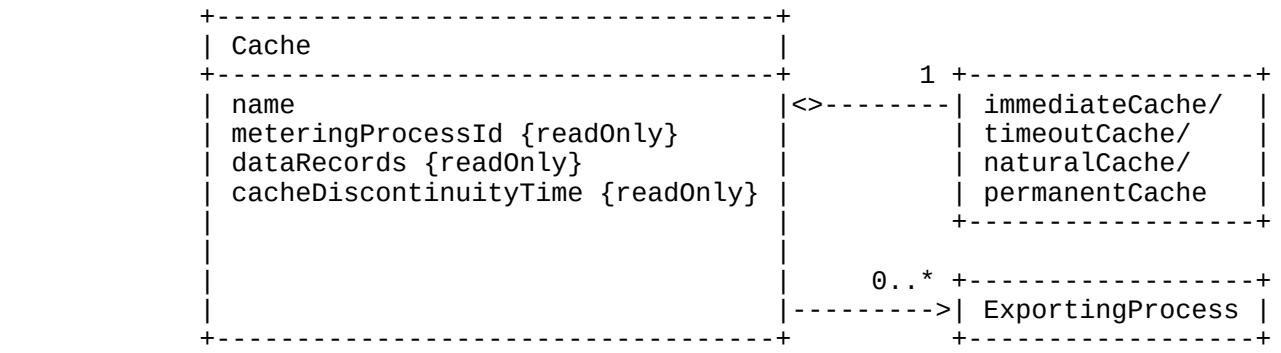


Figure 12: Cache class

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 immediateCache, timeoutCache, naturalCache, and permanentCache, which are presented below in Sections 4.3.1 and 4.3.2. The following three state parameters are common to all Caches and therefore included in the Cache class itself:

meteringProcessId: The identifier of the Metering Process the 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 [RFC5101] with the corresponding object of the MeteringProcess class.

dataRecords: 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 [RFC6615].

cacheDiscontinuityTime: Timestamp of the most recent occasion at which dataRecords suffered a discontinuity. In contrast to ipfixMeteringProcessDiscontinuityTime, 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 ExportingProcess objects configuring different Exporting Processes.

### 4.3.1. ImmediateCache Class

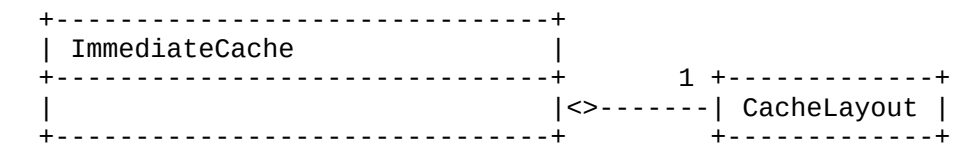


Figure 13: ImmediateCache class

The ImmediateCache 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 CacheLayout class, which is defined below in Section 4.3.3.

### 4.3.2. TimeoutCache, NaturalCache, and PermanentCache Class

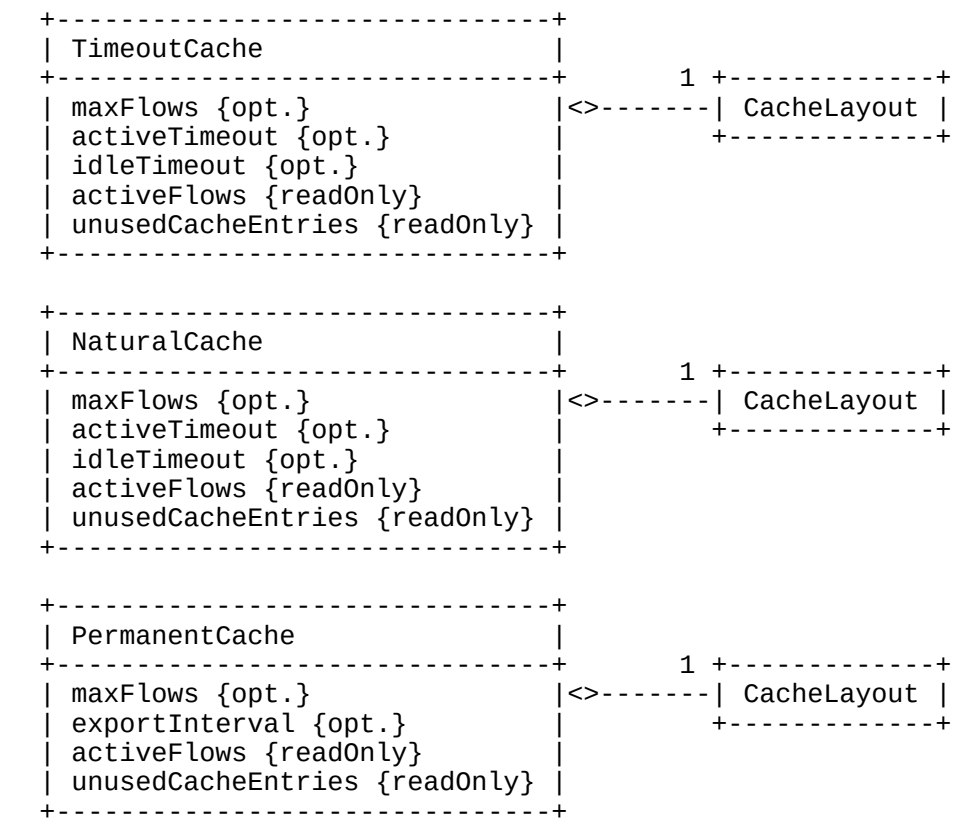


Figure 14: TimeoutCache, NaturalCache, and PermanentCache class

Figure 14 shows the TimeoutCache class, the NaturalCache class, and the PermanentCache class. These classes are used to configure a Cache that aggregates the packets at its input and generates IPFIX Flow Records. The three classes differ in when Flows expire:

- TimeoutCache: Flows expire after active or idle timeout.
- NaturalCache: Flows expire after active or idle timeout, or on natural termination (e.g., TCP FIN or TCP RST) of the Flow.
- PermanentCache: Flows never expire, but are periodically exported with the interval set by exportInterval.

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

maxFlows: 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.

activeFlows: 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].

unusedCacheEntries: The number of unused cache entries. Note that the sum of activeFlows and unusedCacheEntries equals maxFlows if maxFlows is configured.
Note that this parameter corresponds to ipfixMeteringProcessCacheUnusedCacheEntries in the IPFIX MIB module [RFC6615].

The following timeout parameters are only available in the TimeoutCache class and the NaturalCache class:

activeTimeout: 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].

idleTimeout: 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 PermanentCache class:

exportInterval: 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 CacheLayout class that defines which fields are included in the Flow Records.

### 4.3.3. CacheLayout Class

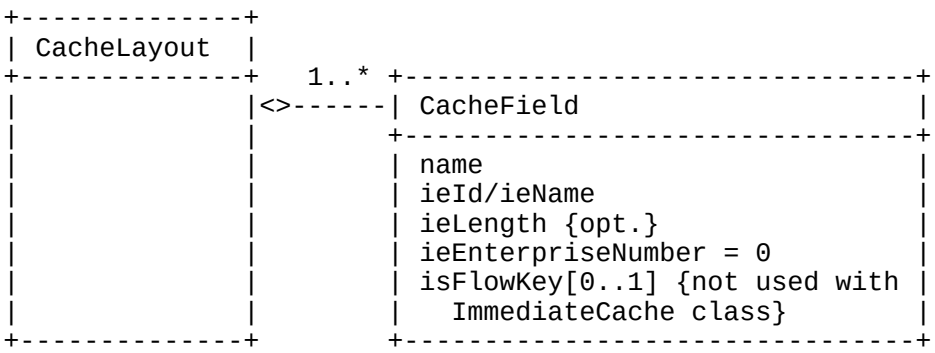


Figure 15: CacheLayout class

A Cache generates and maintains Packet Reports or Flow Records containing information that has been extracted from the incoming stream of packets. Using the CacheField class, the CacheLayout 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 ImmediateCache 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 TimeoutCache, NaturalCache, or PermanentCache 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 [RFC5102] 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.

The CacheLayout class does not have any parameters. The configuration parameters of the CacheField class are as follows:

ieId, ieName, ieEnterpriseNumber: These parameters specify a field by the combination of the Information Element identifier or name, and the Information Element enterprise number. Either ieId or ieName MUST be specified. If ieEnterpriseNumber 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 ieEnterpriseNumber 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].

ieLength: 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 [RFC5101], Section 6.2). If not configured by the user, the field length is set by the Monitoring Device.

isFlowKey: 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 ImmediateCache 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. ExportingProcess Class

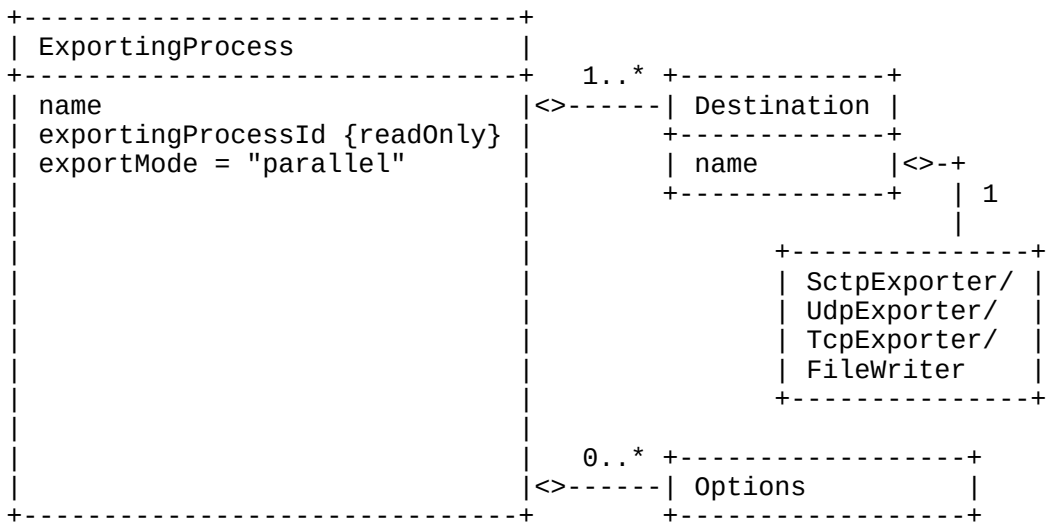


Figure 16: ExportingProcess class

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 one object of the SctpExporter, UdpExporter, TcpExporter, or FileWriter class which contains further configuration parameters. These classes are described in Sections 4.4.1, 4.4.2, 4.4.3, and 4.4.4.

As state parameter, the ExportingProcess class contains the identifier of the Exporting Process (exportingProcessId). 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 [RFC5101] with the corresponding object of the ExportingProcess class.

The order in which objects of the Destination class appear is defined by the user. However, the order has a specific meaning only if the exportMode parameter is set to "fallback". The exportMode parameter is defined as follows:

exportMode: This parameter 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
* loadBalancing: 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 exportMode is set to "fallback", the first object of the Destination class defines the primary destination, the second object of the Destination class 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 exportMode parameter is related to the ipfixExportMemberType object in [RFC6615]. If exportMode is "parallel", the ipfixExportMemberType values of the corresponding entries in ipfixExportTable are set to parallel(3). If exportMode is "loadBalancing", 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 [RFC5101]. The Exporting Process may also export certain fields in a separate Data Record as described in [RFC5476].

#### 4.4.1. SctpExporter Class

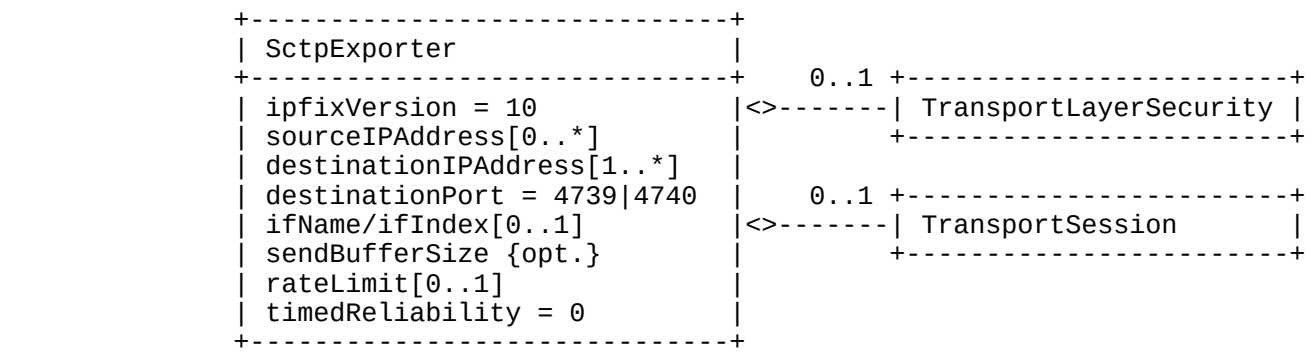


Figure 17: SctpExporter class

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

ipfixVersion: Version number of the IPFIX protocol used. If omitted, the default value is 10 (=0x000a) as specified in [RFC5101].

sourceIPAddress: 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.

destinationIPAddress: 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.

destinationPort: 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.

ifIndex/ifName: Either the index or 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.

sendBufferSize: Size of the socket send buffer in bytes. If not configured by the user, the buffer size is set by the Monitoring Device.

rateLimit: 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.

timedReliability: 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.

If a Transport Session is established to the configured destination, the SctpExporter class includes an object of the TransportSession class containing state parameters of the Transport Session. The TransportSession class is specified in Section 4.7.

### 4.4.2. UdpExporter Class

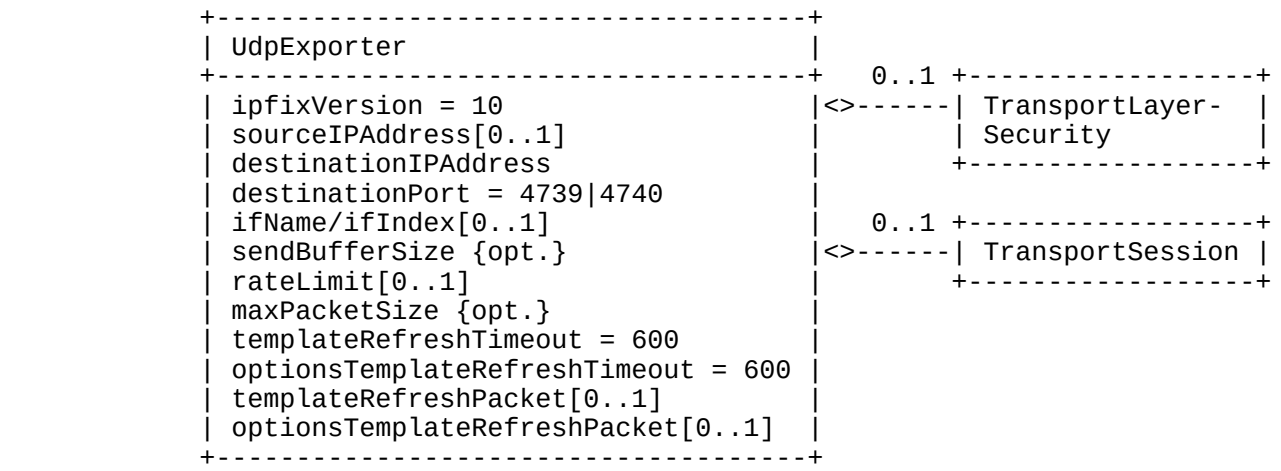


Figure 18: UdpExporter class

The UdpExporter class shown in Figure 18 contains the configuration parameters of a UDP export destination. The parameters ipfixVersion, destinationPort, ifName, ifIndex, sendBufferSize, and rateLimit have the same meaning as in the SctpExporter class (see Section 4.4.1). The remaining configuration parameters are:

sourceIPAddress: 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.

destinationIPAddress: Destination IP address to which IPFIX Messages are sent (i.e., the IP address of the Collecting Process).

maxPacketSize: 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.

templateRefreshTimeout, optionsTemplateRefreshTimeout, templateRefreshPacket, optionsTemplateRefreshPacket: These parameters specify when (Options) Templates are refreshed by the Exporting Process.
templateRefreshTimeout and optionsTemplateRefreshTimeout are specified in seconds between resendings of (Options) Templates.

If omitted, the default value of 600 seconds (10 minutes) is used [RFC5101].
templateRefreshPacket and optionsTemplateRefreshPacket specify the number of IPFIX Messages after which (Options) Templates are resent. If omitted, the (Options) Templates are only resent after timeout.
Note that the values configured for templateRefreshTimeout and optionsTemplateRefreshTimeout MUST be adapted to the templateLifeTime and optionsTemplateLifeTime parameter settings at the receiving Collecting Process (see Section 4.5.2).
Note that these parameters correspond to ipfixTransportSessionTemplateRefreshTimeout, ipfixTransportSessionOptionsTemplateRefreshTimeout, 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 export destination.

If a Transport Session is established to the configured destination, the UdpExporter class includes an object of the TransportSession class containing state parameters of the Transport Session. The TransportSession class is specified in Section 4.7.

### 4.4.3. TcpExporter Class

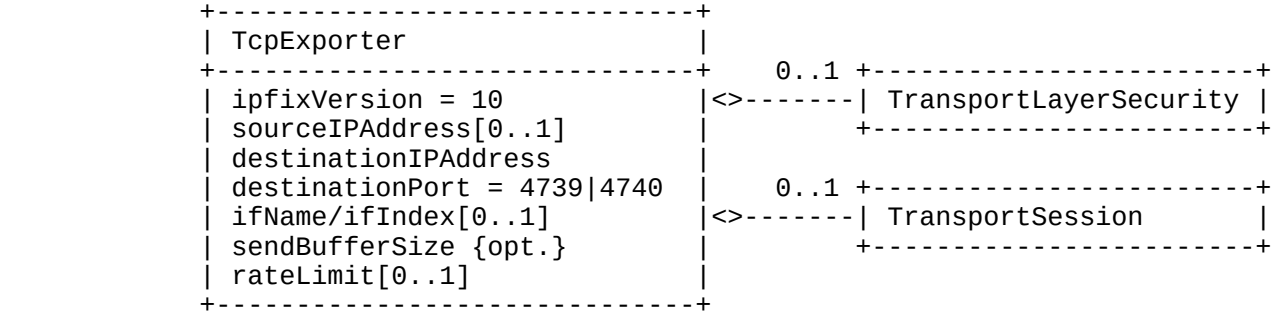


Figure 19: TcpExporter class

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.

If a Transport Session is established to the configured destination, the TcpExporter class includes an object of the TransportSession class containing state parameters of the Transport Session. The TransportSession class is specified in Section 4.7.

### 4.4.4. FileWriter Class

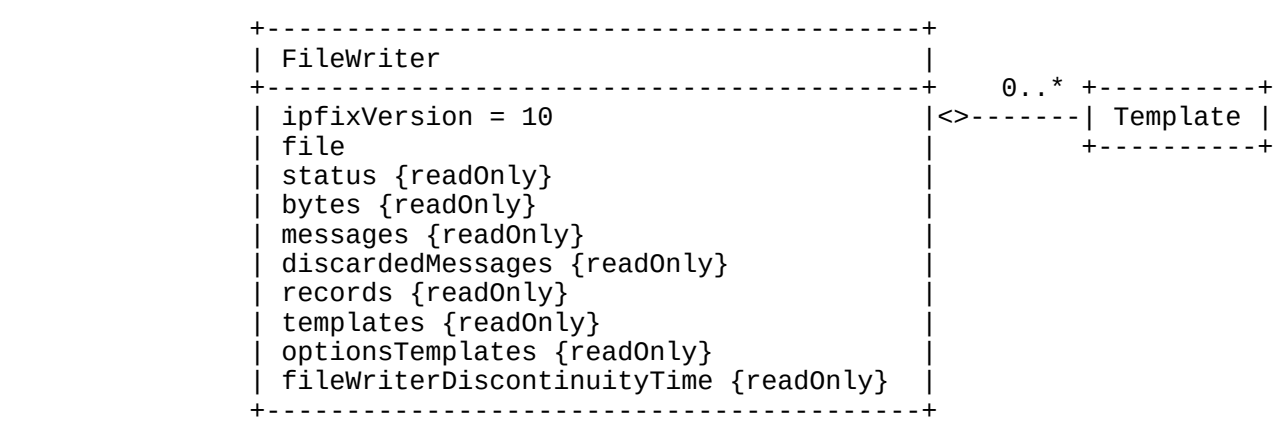


Figure 20: FileWriter classes

If an object of the FileWriter class is included in an object of the Destination class, IPFIX Messages are written into a file as specified in [RFC5655]. The FileWriter class contains the following configuration parameters:

ipfixVersion: Version number of the IPFIX protocol used. If omitted, the default value is 10 (=0x000a) as specified in [RFC5101].

file: File name and location specified as URI.

The state parameters of the FileWriter class are:

bytes, messages, records, templates, optionsTemplates: 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 fileWriterDiscontinuityTime.

discardedMessages: 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 fileWriterDiscontinuityTime.

fileWriterDiscontinuityTime: 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.

Each object of the FileWriter class includes a list of objects of the Template class with information and statistics about the Templates written to the file. The Template class is specified in Section 4.8.

### 4.4.5. Options Class

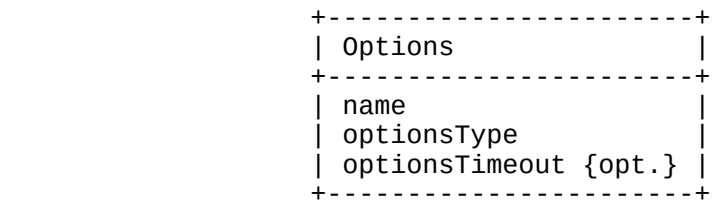


Figure 21: Options class

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. [RFC5101] and [RFC5476] specify several types of reporting information that may be exported. The following parameter values are specified by the configuration data model:

meteringStatistics: Export of Metering Process statistics using the Metering Process Statistics Options Template [RFC5101].

meteringReliability: Export of Metering Process reliability statistics using the Metering Process Reliability Statistics Options Template [RFC5101].

exportingReliability: Export of Exporting Process reliability statistics using the Exporting Process Reliability Statistics Options Template [RFC5101].

flowKeys: Export of the Flow Key specification using the Flow Keys Options Template [RFC5101].

selectionSequence: Export of Selection Sequence Report Interpretation and Selector Report Interpretation [RFC5476].

selectionStatistics: Export of Selection Sequence Statistics Report Interpretation [RFC5476].

accuracy: Export of Accuracy Report Interpretation [RFC5476].

reducingRedundancy: 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.

extendedTypeInformation: 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 optionsTimeout 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 flowKeys, selectionSequence, accuracy, and reducingRedundancy. If optionsTimeout is not configured by the user, it is set by the Monitoring Device.

## 4.5. CollectingProcess Class

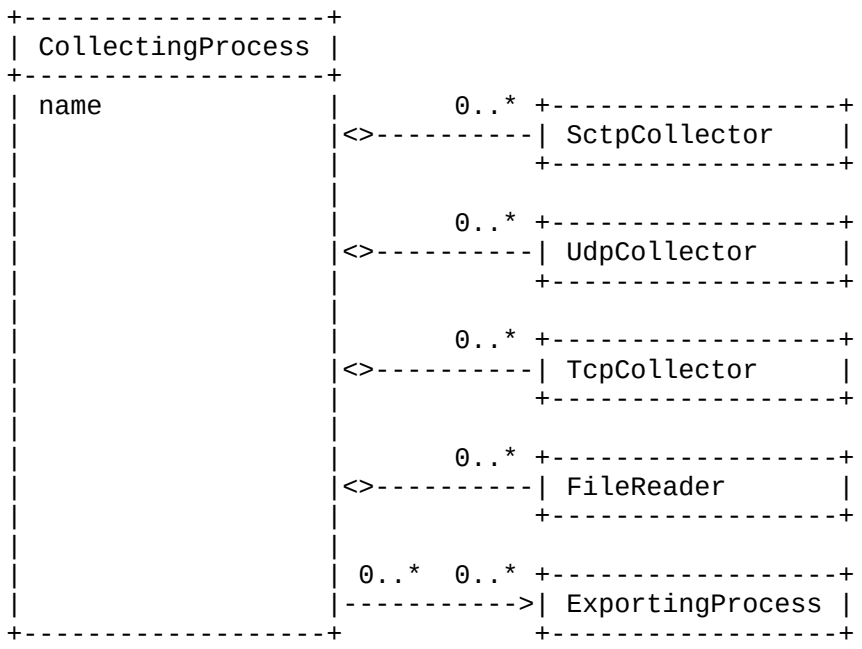


Figure 22: CollectingProcess class

Figure 22 shows the CollectingProcess class that contains the configuration and state parameters of a Collecting Process. Objects of the SctpCollector, UdpCollector, and TcpCollector classes specify how IPFIX Messages are received from remote Exporters. The Collecting Process can also be configured as a File Reader using objects of the FileReader class. These classes are described in Sections 4.5.1, 4.5.2, 4.5.3, and 4.5.4.

A CollectingProcess object MAY refer to one or more ExportingProcess objects configuring Exporting Processes that export the received data without modifications to a file or to another Collector.

#### 4.5.1. SctpCollector Class

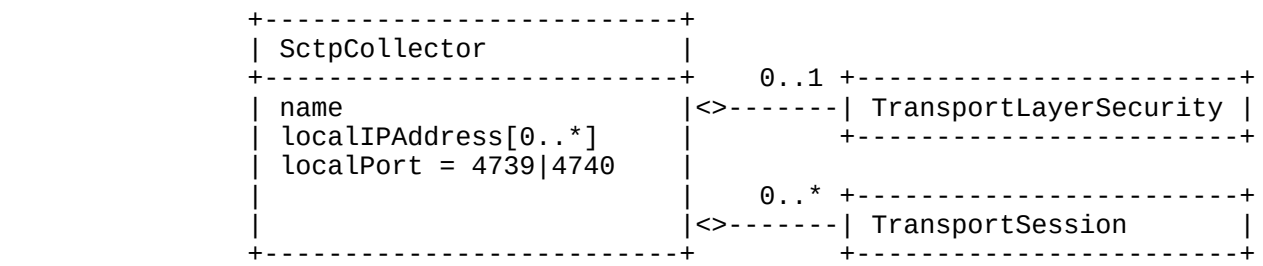


Figure 23: SctpCollector class

The SctpCollector class contains the configuration parameters of a listening SCTP socket at a Collecting Process. The parameters are:

localIPAddress: 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.

localPort: 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.

As state data, the SctpCollector class contains the list of currently established Transport Sessions that terminate at the given SCTP socket of the Collecting Process. The TransportSession class is specified in Section 4.7.

### 4.5.2. UdpCollector Class

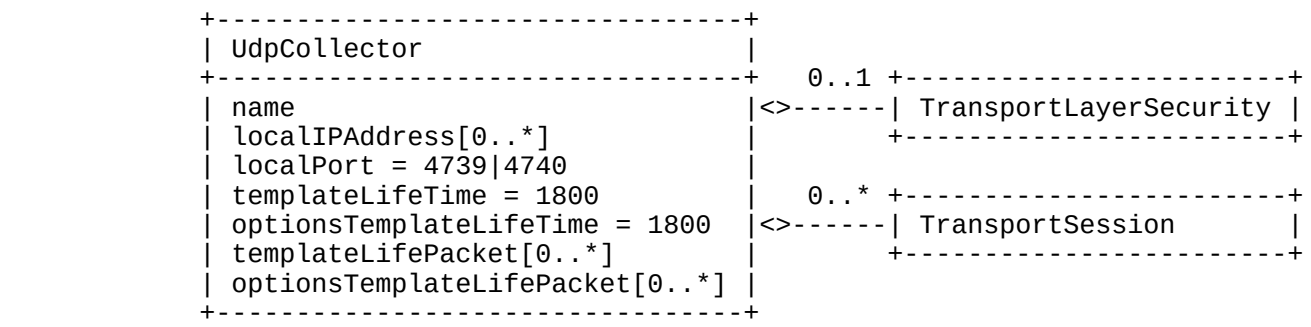


Figure 24: UdpCollector class

The UdpCollector class contains the configuration parameters of a listening UDP socket at a Collecting Process. The parameter localPort has the same meaning as in the SctpCollector class (see Section 4.5.1). The remaining parameters are:

localIPAddress: 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.

templateLifeTime, optionsTemplateLifeTime: (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 [RFC5101], Section 10.3.7, the lifetime of Templates and Options Templates MUST be at least three times higher than the templateRefreshTimeout and optionTemplatesRefreshTimeout 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 [RFC5101].
Note that these parameters correspond to ipfixTransportSessionTemplateRefreshTimeout and ipfixTransportSessionOptionsTemplateRefreshTimeout in the IPFIX MIB module [RFC6615].

templateLifePacket, optionsTemplateLifePacket: If templateLifePacket 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 templateLifeTime. Similarly, if

optionsTemplateLifePacket 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 optionsTemplateLifeTime.
If not configured, Templates and Options Templates only become invalid according to the lifetimes specified by templateLifeTime and optionsTemplateLifeTime, 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.

As state data, the UdpCollector class contains the list of currently established Transport Sessions that terminate at the given UDP socket of the Collecting Process. The TransportSession class is specified in Section 4.7.

### 4.5.3. TcpCollector Class

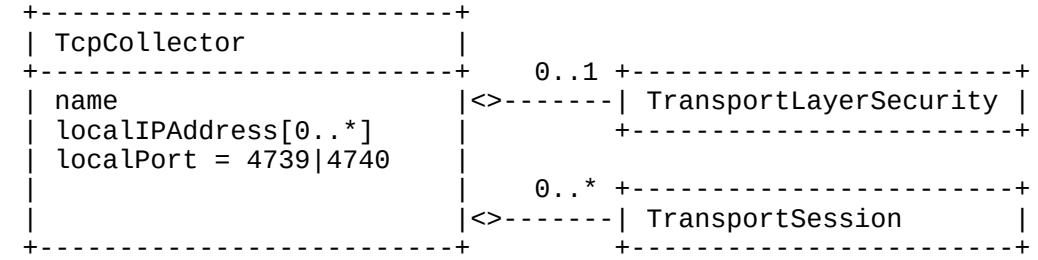


Figure 25: TcpCollector class

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 (see Section 4.5.2).

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

As state data, the TcpCollector class contains the list of currently established Transport Sessions that terminate at the given TCP socket of the Collecting Process. The TransportSession class is specified in Section 4.7.

### 4.5.4. FileReader Class

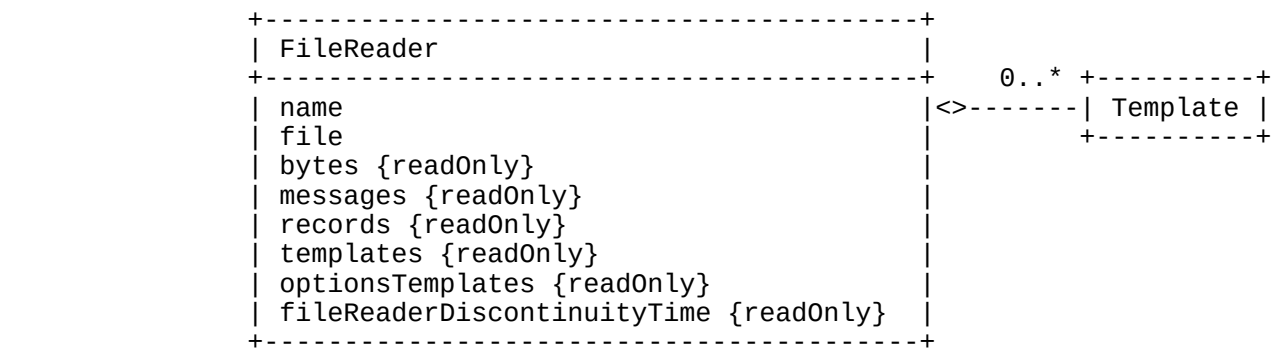


Figure 26: FileReader classes

The Collecting Process may import IPFIX Messages from a file as specified in [RFC5655]. 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, optionsTemplates: 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 fileReaderDiscontinuityTime.

fileReaderDiscontinuityTime: 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.

Each object of the FileReader class includes a list of objects of the Template class with information and statistics about the Templates read from the file. The Template class is specified in Section 4.8.

### 4.6. Transport Layer Security Class

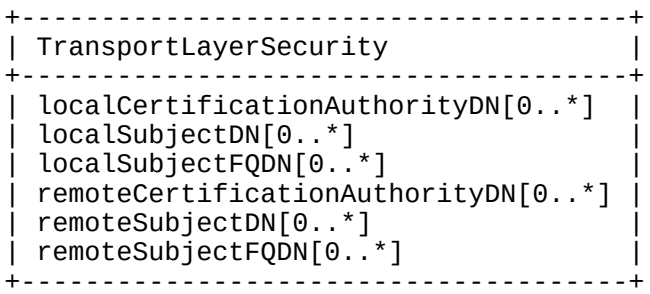


Figure 27: TransportLayerSecurity class

The TransportLayerSecurity class is used in the Exporting Process's SctpExporter, UdpExporter, and TcpExporter classes, and the Collecting Process's SctpCollector, UdpCollector, and TcpCollector classes to enable and configure TLS/DTLS for IPFIX. TLS/DTLS can be enabled without configuring any additional parameters. In this case, an empty XML element <transportLayerSecurity /> appears in the configuration. If TLS/DTLS is enabled, the endpoint must use DTLS [RFC6347] if the transport protocol is SCTP or UDP, and TLS [RFC5246] if the transport protocol is TCP.

[RFC5101] mandates strong mutual authentication of Exporting Processes and Collecting Process as follows. Note this text cites [RFC3280], which was obsoleted by [RFC5280].

> 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 [RFC3280]. 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.

The configuration parameters are defined as follows:

localCertificationAuthorityDN: 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 localCertificationAuthorityDN is not configured, the choice of certificates identifying the local endpoint is not restricted with respect to the issuing certification authority.

localSubjectDN, localSubjectFQDN: 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 localSubjectDN contains a distinguished name identifying the local endpoint. Each occurrence of localSubjectFQDN 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 localSubjectDN and localSubjectFQDN is configured at the same time as the localCertificationAuthorityDN parameter, certificates MUST also fulfill the specified restrictions regarding the certification authority.
If localSubjectDN and localSubjectFQDN are not configured, the choice of certificates identifying the local endpoint is not restricted with respect to the subject's distinguished name or FQDN.

remoteCertificationAuthorityDN: 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 remoteCertificationAuthorityDN is not configured, the authorization of remote endpoints is not restricted with respect to the issuing certification authority of the delivered certificate.

remoteSubjectDN, remoteSubjectFQDN: 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 remoteSubjectDN contains a distinguished name identifying a remote endpoint. Each occurrence of remoteSubjectFQDN 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 commonName component of the subject field. Certificates not

fulfilling this condition MUST be rejected during TLS/DTLS handshake.
If any of the parameters remoteSubjectDN and remoteSubjectFQDN is configured at the same time as the remoteCertificationAuthorityDN parameter, certificates MUST also fulfill the specified restrictions regarding the certification authority in order to be accepted.
If remoteSubjectDN and remoteSubjectFQDN 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

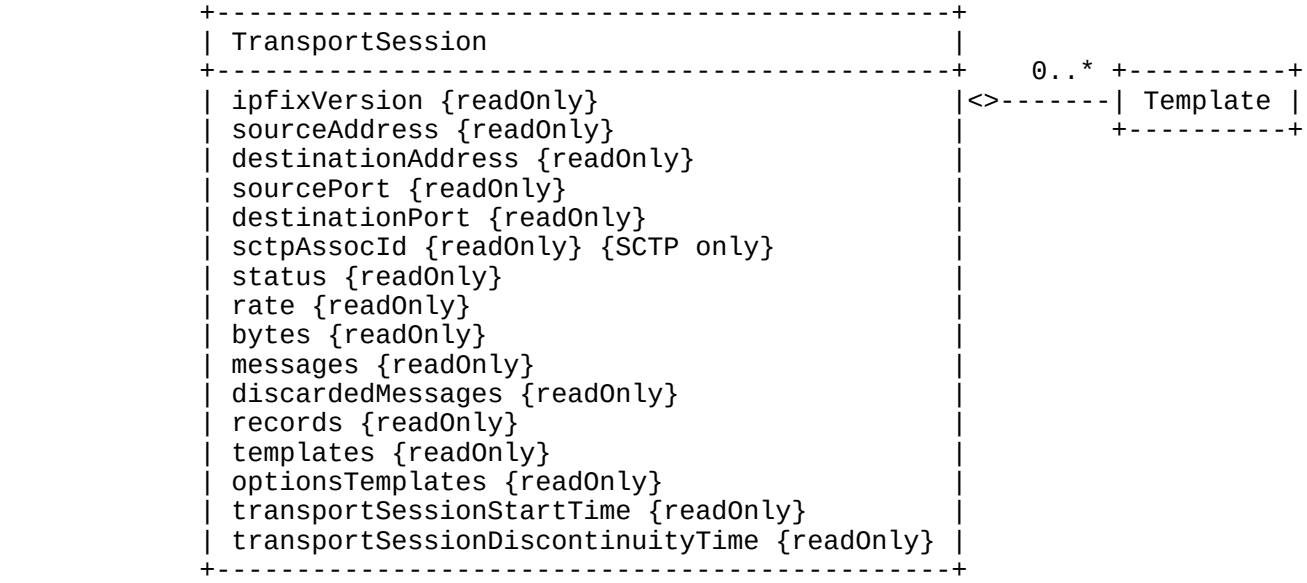


Figure 28: TransportSession class

The TransportSession class contains state data about Transport Sessions originating from an Exporting Process or terminating at a Collecting Process. In general, the state parameters correspond to the managed objects in the ipfixTransportSessionTable and ipfixTransportSessionStatsTable of the IPFIX MIB module [RFC6615]. An exception is the usage of the parameters sourceAddress and destinationAddress. If SCTP is the transport protocol, the Exporter or Collector MAY be multihomed SCTP endpoints (see [RFC4960], Section 6.4) and use more than one IP address. In the IPFIX MIB module, ipfixTransportSessionSctpAssocId is used instead of ipfixTransportSessionSourceAddress and

ipfixTransportSessionDestinationAddress to point to an entry in the sctpAssocTable defined in the SCTP MIB module [RFC3871]. Since we cannot assume that an SNMP agent offering access to the SCTP MIB module exists on the Monitoring Device, the configuration data model cannot rely on this parameter. Therefore, the state parameters sourceAddress and destinationAddress are used for SCTP as well, containing one of the potentially many Exporter and Collector IP addresses in the SCTP association. Preferably, the IP addresses of the path that is usually selected by the Exporter to send IPFIX Messages to the Collector SHOULD be contained.

Several MIB objects of the ipfixTransportSessionTable are omitted in the TransportSession class. The MIB object ipfixTransportSessionDeviceMode is not included because its value can be derived from the context in which a TransportSession object appears: exporting(1) if it belongs to an Exporting Process, collecting(2) if it belongs to a Collecting Process. Similarly, the MIB object ipfixTransportSessionProtocol is not included as the transport protocol is known from the context as well. The MIB objects ipfixTransportSessionTemplateRefreshTimeout, ipfixTransportSessionOptionsTemplateRefreshTimeout, ipfixTransportSessionTemplateRefreshPacket, and ipfixTransportSessionOptionsTemplateRefreshPacket are not included since they correspond to configuration parameters of the UdpExporter class (templateRefreshTimeout, optionsTemplateRefreshTimeout, templateRefreshPacket, optionsTemplateRefreshPacket) and the UdpCollector class (templateLifeTime, optionsTemplateLifeTime, templateLifePacket, optionsTemplateLifePacket).

ipfixVersion: 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 ipfixVersion of the outer SctpExporter, UdpExporter, or TcpExporter object.
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].

sourceAddress, destinationAddress: If TCP or UDP is the transport protocol, sourceAddress contains the IP address of the Exporter, and destinationAddress 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, sourceAddress contains one of the IP addresses of the Exporter and destinationAddress 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.

sourcePort, destinationPort: 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].

sctpAssocId: 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, optionsTemplates: 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 transportSessionDiscontinuityTime.

discardedMessages: 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 transportSessionDiscontinuityTime.

transportSessionStartTime: Timestamp of the start of the given Transport Session.
This state parameter does not correspond to any object in the IPFIX MIB module.

transportSessionDiscontinuityTime: Timestamp of the most recent occasion at which one or more of the Transport Session counters suffered a discontinuity. In contrast to ipfixTransportSessionDiscontinuityTime, the time is absolute and not relative to sysUpTime.

Note that, if used for Exporting Processes, the values of the state parameters destinationAddress and destinationPort match the values of the configuration parameters destinationIPAddress and destinationPort of the outer SctpExporter, TcpExporter, and UdpExporter objects (in the case of SctpExporter, one of the configured destinationIPAddress values); if the transport protocol is UDP or SCTP and if the parameter sourceIPAddress is configured in the outer UdpExporter or SctpExporter object, the value of sourceAddress equals the configured value or one of the configured values. Used for Collecting Processes, the value of destinationAddress equals the value (or one of the values) of the parameter localIPAddress if this parameter is configured in the outer UdpCollector, TcpCollector, or SctpCollector object; destinationPort equals the value of the configuration parameter localPort.

Each object of the TransportSession class includes a list of objects of the Template class with information and statistics about the Templates transmitted or received on the given Transport Session. The Template class is specified in Section 4.8.

## 4.8. Template Class

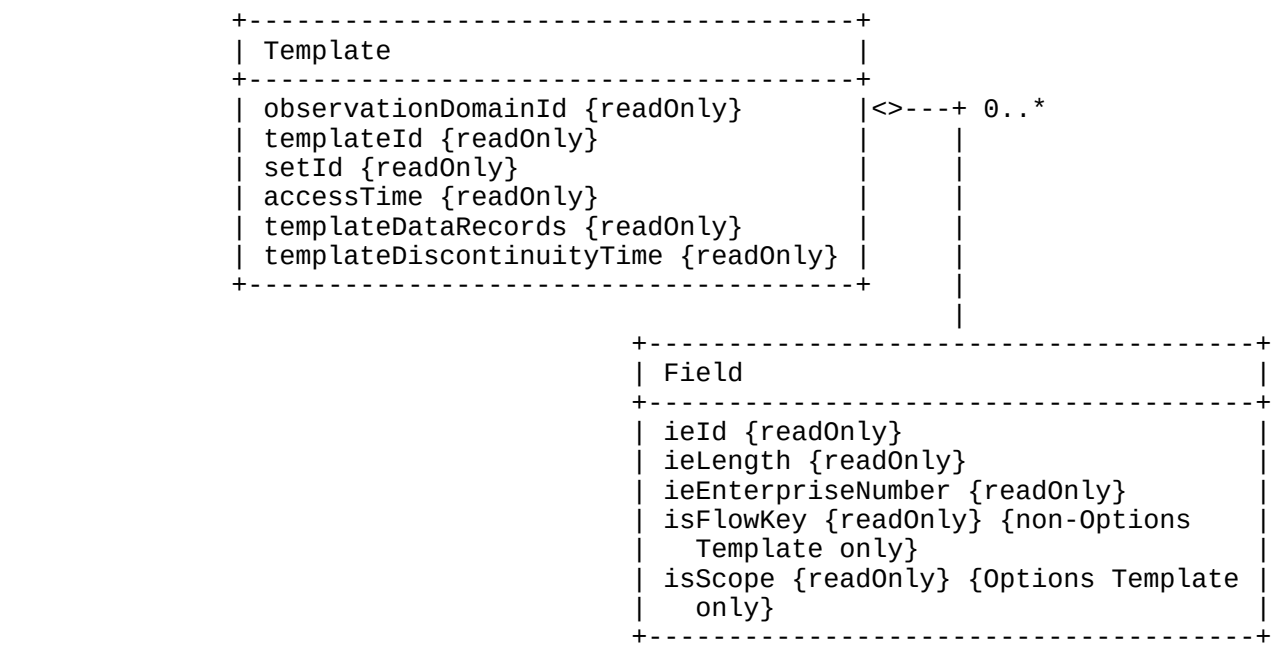


Figure 29: Template class

The Template class 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. 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]:

observationDomainId: The ID of the Observation Domain for which this Template is defined.

templateId: This number indicates the Template ID in the IPFIX Message.

setId: This number indicates the Set ID of the Template.
Currently, there are two values defined [RFC5101]. The value 2 is used for Sets containing Template definitions. The value 3 is used for Sets containing Options Template definitions.

accessTime: 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.

templateDataRecords: The number of transmitted or received Data Records defined by this (Options) Template since the point in time indicated by templateDefinitionTime.

templateDiscontinuityTime: Timestamp of the most recent occasion at which the counter templateDataRecords suffered a discontinuity. In contrast to ipfixTemplateDiscontinuityTime, the time is absolute and not relative to sysUpTime.

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

isFlowKey: 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).

isFlowKey: 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).

## 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.

YANG [RFC6020] offers several possibilities to restrict and adapt a configuration data model. The current version of YANG defines the concepts of features, deviations, and extensions.

The feature concept allows the author of a configuration data model to make proportions of the model conditional in a manner that is controlled by the device. Devices do not have to support these conditional parts to conform to the model. If the NETCONF protocol is used, features which are supported by the device are announced in the <hello> message [RFC6241].

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 proportions 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 proportions of the model are conditional on the following feature:

meter: If this feature is supported, Observation Points, Selection Processes, and Caches can be configured.

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

psampSampCountBased: If this feature is supported, Sampling method sampCountBased can be configured.

psampSampTimeBased: If this feature is supported, Sampling method sampTimeBased can be configured.

psampSampRandOutOfN: If this feature is supported, Sampling method sampRandOutOfN can be configured.

psampSampUniProb: If this feature is supported, Sampling method sampUniProb can be configured.

psampFilterMatch: If this feature is supported, Filtering method filterMatch can be configured.

psampFilterHash: If this feature is supported, Filtering method filterHash can be configured.

immediateCache: If this feature is supported, a Cache generating PSAMP Packet Reports can be configured using the ImmediateCache class.

timeoutCache: If this feature is supported, a Cache generating IPFIX Flow Records can be configured using the TimeoutCache class.

naturalCache: If this feature is supported, a Cache generating IPFIX Flow Records can be configured using the NaturalCache class.

permanentCache: If this feature is supported, a Cache generating IPFIX Flow Records can be configured using the PermanentCache class.

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

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

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

fileReader: If this feature is supported, File Readers can be configured as part of Collecting Processes.

fileWriter: If this feature is supported, File Writers can be configured as part of Exporting Processes.

The deviation concept enables a device to announce deviations from the standard model using the "deviation" statement. For example, it is possible to restrict the value range of a specific parameter or to define that the configuration of a certain parameter is not supported at all. Hence, deviations are typically used to specify limitations

due to resource constraints or functional restrictions. Deviations concern existing parameters of the original configuration data model and must not be confused with model extensions. Model extensions are specified with the "augment" statement and allow adding new parameters to the original configuration data model.

If certain device-specific constraints cannot be formally specified with YANG, they MUST be expressed with human-readable text using the "description" statement. The provided information MUST enable the user to define a configuration that is entirely supported by the Monitoring Device. On the other hand, if a Monitoring Device is configured, it MUST notify the user about any part of the configuration that is not supported. The Monitoring Device MUST NOT silently accept configuration data that cannot be completely enforced. If the NETCONF protocol is used to send configuration data to the Monitoring Device, the error handling is specified in the NETCONF protocol specification [RFC6241].

Just like features, deviations and model extensions are announced in NETCONF's <hello> message. A usage example of deviations is given in Section 7.5.

## 6. YANG Module of the IPFIX/PSAMP Configuration Data Model

The YANG module specification of the configuration data model is listed below. It makes use of the common YANG types defined in the modules urn:ietf:params:xml:ns:yang:ietf-yang-types and urn:ietf:params:xml:ns:yang:ietf-inet-types [RFC6021].

```
<CODE BEGINS> file "ietf-ipfix-psamp@2012-09-05.yang"
module ietf-ipfix-psamp {
  namespace "urn:ietf:params:xml:ns:yang:ietf-ipfix-psamp";
  prefix ipfix;

  import ietf-yang-types { prefix yang; }
  import ietf-inet-types { prefix inet; }

  organization
    "IETF IPFIX Working Group";

  contact
    "WG Web:  <http://tools.ietf.org/wg/ipfix/>
    WG List:  <ipfix@ietf.org>

    WG Chair: Nevil Brownlee
              <n.brownlee@auckland.ac.nz>
```

```
      WG Chair: Juergen Quittek
                <quittek@neclab.eu>

      Editor:   Gerhard Muenz
                <muenz@net.in.tum.de>";

    description
      "IPFIX/PSAMP Configuration Data Model

      Copyright (c) 2012 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).";

    revision 2012-09-05 {
      description "Initial version";
      reference "RFC 6728: Configuration Data Model for the IP Flow
        Information Export (IPFIX) and Packet Sampling (PSAMP)
        Protocols";
    }

    /*****************************************************************
    * Features
    *****************************************************************/

    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 meter {
      description "If supported, Observation Points, Selection
        Processes, and Caches can be configured.";
    }

    feature psampSampCountBased {
      description "If supported, the Monitoring Device supports
        count-based Sampling.  The Selector method sampCountBased can
        be configured.";
```

```
    }

    feature psampSampTimeBased {
      description "If supported, the Monitoring Device supports
        time-based Sampling.  The Selector method sampTimeBased can
        be configured.";
    }

    feature psampSampRandOutOfN {
      description "If supported, the Monitoring Device supports
        random n-out-of-N Sampling.  The Selector method
        sampRandOutOfN can be configured.";
    }

    feature psampSampUniProb {
      description "If supported, the Monitoring Device supports
        uniform probabilistic Sampling.  The Selector method
        sampUniProb can be configured.";
    }

    feature psampFilterMatch {
      description "If supported, the Monitoring Device supports
        property match Filtering.  The Selector method filterMatch
        can be configured.";
    }

    feature psampFilterHash {
      description "If supported, the Monitoring Device supports
        hash-based Filtering.  The Selector method filterHash can be
        configured.";
    }

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

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

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

```
    }

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

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

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

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

    feature fileWriter {
      description "If supported, the Monitoring Device supports the
        configuration of Exporting Processes as File Writers.";
    }

    /*****************************************************************
    * Identities
    *****************************************************************/

    /*** Hash function identities ***/
    identity hashFunction {
      description "Base identity for all hash functions used for
        hash-based packet Filtering.  Identities derived from
        this base are used by the leaf
        /ipfix/selectionProcess/selector/filterHash/hashFunction.";
    }
    identity BOB {
      base "hashFunction";
      description "BOB hash function";
      reference "RFC 5475, Section 6.2.4.1.";
    }
    identity IPSX {
      base "hashFunction";
      description "IPSX hash function";
      reference "RFC 5475, Section 6.2.4.1.";
```

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

    /*** Export mode identities ***/
    identity exportMode {
      description "Base identity for different usages of export
        destinations configured for an Exporting Process.
        Identities derived from this base are used by the leaf
        /ipfix/exportingProcess/exportMode.";
    }
    identity parallel {
      base "exportMode";
      description "Parallel export of Data Records to all
        destinations configured for the Exporting Process.";
    }
    identity loadBalancing {
      base "exportMode";
      description "Load-balancing between the different destinations
        configured for the Exporting Process.";
    }
    identity fallback {
      base "exportMode";
      description "Export to the primary destination (i.e., the first
        SCTP, UDP, TCP, or file 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.";
    }

    /*** Options type identities ***/
    identity optionsType {
      description "Base identity for report types exported with
        options.  Identities derived from this base are used by the leaf
        /ipfix/exportingProcess/options/optionsType.";
    }
    identity meteringStatistics {
      base "optionsType";
      description "Metering Process Statistics.";
      reference "RFC 5101, Section 4.1.";
    }
    identity meteringReliability {
      base "optionsType";
      description "Metering Process Reliability Statistics.";
```

```
      reference "RFC 5101, Section 4.2.";
    }
    identity exportingReliability {
      base "optionsType";
      description "Exporting Process Reliability
        Statistics.";
      reference "RFC 5101, Section 4.3.";
    }
    identity flowKeys {
      base "optionsType";
      description "Flow Keys.";
      reference "RFC 5101, Section 4.4.";
    }
    identity selectionSequence {
      base "optionsType";
      description "Selection Sequence and Selector Reports.";
      reference "RFC 5476, Sections 6.5.1 and 6.5.2.";
    }
    identity selectionStatistics {
      base "optionsType";
      description "Selection Sequence Statistics Report.";
      reference "RFC 5476, Sections 6.5.3.";
    }
    identity accuracy {
      base "optionsType";
      description "Accuracy Report.";
      reference "RFC 5476, Section 6.5.4.";
    }
    identity reducingRedundancy {
      base "optionsType";
      description "Enables the utilization of Options Templates to
        reduce redundancy in the exported Data Records.";
      reference "RFC 5473.";
    }
    identity extendedTypeInformation {
      base "optionsType";
      description "Export of extended type information for
        enterprise-specific Information Elements used in the
        exported Templates.";
      reference "RFC 5610.";
    }

    /****************************************************************
    * Type definitions
    ****************************************************************/

    typedef ieNameType {
      type string {
```

```
        length "1..max";
        pattern "\S+";
      }
      description "Type for Information Element names.  Whitespaces
        are not allowed.";
    }

    typedef ieIdType {
      type uint16 {
        range "1..32767" {
          description "Valid range of Information Element
              identifiers.";
          reference "RFC 5102, Section 4.";
        }
      }
      description "Type for Information Element identifiers.";
    }

    typedef nameType {
      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 ifNameType {
      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 {
          description "This value is used for monitoring incoming
            packets.";
        }
        enum egress {
          description "This value is used for monitoring outgoing
            packets.";
        }
```

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

    typedef transportSessionStatus {
      type enumeration {
        enum inactive {
          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 {
          description "This value MUST be used for Transport Sessions
            that are currently active and transmitting or receiving
            data.";
        }
        enum unknown {
          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).";
    }

    /*****************************************************************
    * Groupings
    *****************************************************************/

    grouping observationPointParameters {
      description "Interface as input to Observation Point.";
      leaf observationPointId {
        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.";
      }
      leaf observationDomainId {
```

```
        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 5101; RFC 6615, Section 8
          (ipfixObservationPointObservationDomainId).";
      }
      leaf-list ifName {
        type ifNameType;
        description "List of names identifying interfaces of the
          Monitoring Device.  The Observation Point observes packets at
          the specified interfaces.";
      }
      leaf-list ifIndex {
        type uint32;
        description "List of ifIndex 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 entPhysicalName {
        type string;
        description "List of names identifying physical entities of the
          Monitoring Device.  The Observation Point observes packets at
          the specified entities.";
      }
      leaf-list entPhysicalIndex {
        type uint32;
        description "List of entPhysicalIndex 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 4133; 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 sampCountBasedParameters {
      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 packetInterval {
        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 packetSpace {
        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 sampTimeBasedParameters {
      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 timeInterval {
        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 timeSpace {
        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 sampRandOutOfNParameters {
      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 sampUniProbParameters {
      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 filterMatchParameters {
      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 nameOrId {
        mandatory true;
        description "The field to be matched is specified by
          either the name or the identifier of the Information
          Element.";
        leaf ieName {
          type ieNameType;
          description "Name of the Information Element.";
```

```
        }
        leaf ieId {
          type ieIdType;
          description "Identifier of the Information Element.";
        }
      }
      leaf ieEnterpriseNumber {
        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 filterHashParameters {
      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 hashFunction {
        type identityref {
          base "hashFunction";
        }
        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 initializerValue {
        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 ipPayloadOffset {
        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 ipPayloadSize {
        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 digestOutput {
        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.";
      }
      leaf outputRangeMin {
        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 outputRangeMax {
        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).";
      }
      list selectedRange {
        key name;
        min-elements 1;
        description "List of hash function return ranges for
          which packets are selected.";
        leaf name {
          type nameType;
          description "Key of this list.";
        }
        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 selectorParameters {
      description "Configuration and state parameters of a Selector.";
      choice Method {
        mandatory true;
        description "Packet selection method applied by the Selector.";
        leaf selectAll {
          type empty;
          description "Method that selects all packets.";
        }
        container sampCountBased {
          if-feature psampSampCountBased;
          description "Systematic count-based packet Sampling.";
          uses sampCountBasedParameters;
        }
        container sampTimeBased {
          if-feature psampSampTimeBased;
          description "Systematic time-based packet Sampling.";
          uses sampTimeBasedParameters;
        }
        container sampRandOutOfN {
          if-feature psampSampRandOutOfN;
          description "n-out-of-N packet Sampling.";
          uses sampRandOutOfNParameters;
        }
        container sampUniProb {
          if-feature psampSampUniProb;
          description "Uniform probabilistic packet Sampling.";
          uses sampUniProbParameters;
        }
        container filterMatch {
          if-feature psampFilterMatch;
          description "Property match Filtering.";
          uses filterMatchParameters;
        }
        container filterHash {
          if-feature psampFilterHash;
          description "Hash-based Filtering.";
          uses filterHashParameters;
        }
      }
      leaf packetsObserved {
        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 packetsDropped {
        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 selectorDiscontinuityTime {
        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 cacheLayoutParameters {
      description "Cache Layout parameters used by immediateCache,
        timeoutCache, naturalCache, and permanentCache.";
      container cacheLayout {
        description "Cache Layout parameters.";
        list cacheField {
          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 nameType;
            description "Key of this list.";
          }
          choice nameOrId {
            mandatory true;
            description "Name or identifier of the Information
              Element.";
            reference "RFC 5102, Section 2; IANA registry for IPFIX
              Entities, http://www.iana.org/assignments/ipfix.";
            leaf ieName {
              type ieNameType;
              description "Name of the Information Element.";
            }
            leaf ieId {
              type ieIdType;
              description "Identifier of the Information Element.";
            }
          }
          leaf ieLength {
            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 5101, Section 6.2.";
          }
          leaf ieEnterpriseNumber {
            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 5101; 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.";
        }
        leaf isFlowKey {
          when "(name(../../..) != 'immediateCache')
            and
            ((count(../ieEnterpriseNumber) = 0)
            or
            (../ieEnterpriseNumber != 29305))" {
            description "This parameter is not available for
              Reverse Information Elements (which have enterprise
              number 29305).  It is also not available for
              immediateCache.";
          }
          type empty;
          description "If present, this is a flow key.";
        }
      }
    }
  }

  grouping flowCacheParameters {
    description "Configuration and state parameters of a Cache
      generating Flow Records.";
    leaf maxFlows {
      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.";
```

```
      }
      leaf activeTimeout {
        when "(name(..) = 'timeoutCache') or
          (name(..) = 'naturalCache')" {
          description "This parameter is only available for
            timeoutCache and naturalCache.";
        }
        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 idleTimeout {
        when "(name(..) = 'timeoutCache') or
          (name(..) = 'naturalCache')" {
          description "This parameter is only available for
            timeoutCache and naturalCache.";
        }
        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).";
      }
      leaf exportInterval {
        when "name(..) = 'permanentCache'" {
          description "This parameter is only available for
            permanentCache.";
        }
```

```
        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.";
      }
      leaf activeFlows {
        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 unusedCacheEntries {
        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).";
      }
    }

    grouping exportingProcessParameters {
      description "Parameters of an Exporting Process.";
      leaf exportingProcessId {
        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 5101, Section 4.3; IANA registry for IPFIX
          Entities, http://www.iana.org/assignments/ipfix.";
      }
```

```
      leaf exportMode {
        type identityref {
          base "exportMode";
        }
        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 nameType;
          description "Key of this list.";
        }
        choice DestinationParameters {
          mandatory true;
          description "Configuration parameters depend on whether
            SCTP, UDP, or TCP is used as transport protocol, and
            whether the destination is a file.";
          container sctpExporter {
            description "SCTP parameters.";
            uses sctpExporterParameters;
          }
          container udpExporter {
            if-feature udpTransport;
            description "UDP parameters.";
            uses udpExporterParameters;
          }
          container tcpExporter {
            if-feature tcpTransport;
            description "TCP parameters.";
            uses tcpExporterParameters;
          }
          container fileWriter {
            if-feature fileWriter;
            description "File Writer parameters.";
            uses fileWriterParameters;
          }
        }
      }
      list options {
        key name;
        description "List of options reported by the Exporting
          Process.";
        leaf name {
          type nameType;
```

```
          description "Key of this list.";
        }
        uses optionsParameters;
      }
    }

    grouping commonExporterParameters {
      description "Parameters of en export destination that are
        common to all transport protocols.";
      leaf ipfixVersion {
        type uint16;
        default 10;
        description "IPFIX version number.";
        reference "RFC 5101.";
      }
      leaf destinationPort {
        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.";
      }
      choice indexOrName {
        description "Index or name of the interface as stored in the
          ifTable of IF-MIB.
          If configured, the Exporting Process MUST use the given
          interface to export IPFIX Messages to the export
          destination.
          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.";
        reference "RFC 2863.";
        leaf ifIndex {
          type uint32;
          description "Index of an interface as stored in the ifTable
            of IF-MIB.";
          reference "RFC 2863.";
        }
        leaf ifName {
          type string;
          description "Name of an interface as stored in the ifTable
            of IF-MIB.";
          reference "RFC 2863.";
        }
      }
      leaf sendBufferSize {
        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 rateLimit {
        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.";
      }
      container transportLayerSecurity {
        presence "If transportLayerSecurity is present, DTLS is
          enabled if the transport protocol is SCTP or UDP, and TLS
          is enabled if the transport protocol is TCP.";
        description "TLS or DTLS configuration.";
        uses transportLayerSecurityParameters;
      }
      container transportSession {
        config false;
        description "State parameters of the Transport Session
          directed to the given destination.";
        uses transportSessionParameters;
      }
    }

    grouping sctpExporterParameters {
      description "SCTP-specific export destination parameters.";
      uses commonExporterParameters;
      leaf-list sourceIPAddress {
        type inet:ip-address;
        description "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.";
        reference "RFC 4960, Section 6.4.";
      }
      leaf-list destinationIPAddress {
        type inet:ip-address;
        min-elements 1;
        description "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.";
        reference "RFC 4960, Section 6.4.";
      }
      leaf timedReliability {
        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.";
      }
    }

    grouping udpExporterParameters {
      description "Parameters of a UDP export destination.";
      uses commonExporterParameters;
      leaf sourceIPAddress {
        type inet:ip-address;
        description "Source IP address used by the Exporting Process.
        If not configured, the IP address assigned to the outgoing
        interface is used as source IP address.";
      }
      leaf destinationIPAddress {
        type inet:ip-address;
        mandatory true;
        description "IP address of the Collection Process to which
          IPFIX Messages are sent.";
      }
      leaf maxPacketSize {
        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 templateRefreshTimeout {
```

```
        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
          templateLifeTime parameter value at the receiving Collecting
          Process.
          Note that this parameter corresponds to
          ipfixTransportSessionTemplateRefreshTimeout in the IPFIX
          MIB module.";
        reference "RFC 5101, Section 10.3.6; RFC 6615, Section 8
          (ipfixTransportSessionTemplateRefreshTimeout).";
      }
      leaf optionsTemplateRefreshTimeout {
        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
          optionsTemplateLifeTime parameter value at the receiving
          Collecting Process.
          Note that this parameter corresponds to
          ipfixTransportSessionOptionsTemplateRefreshTimeout in the
          IPFIX MIB module.";
        reference "RFC 5101, Section 10.3.6; RFC 6615, Section 8
          (ipfixTransportSessionOptionsTemplateRefreshTimeout).";
      }
      leaf templateRefreshPacket {
        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 5101, Section 10.3.6; RFC 6615, Section 8
          (ipfixTransportSessionTemplateRefreshPacket).";
      }
      leaf optionsTemplateRefreshPacket {
        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 5101, Section 10.3.6; RFC 6615, Section 8
          (ipfixTransportSessionOptionsTemplateRefreshPacket).";
      }
    }

    grouping tcpExporterParameters {
      description "Parameters of a TCP export destination.";
      uses commonExporterParameters;
      leaf sourceIPAddress {
        type inet:ip-address;
        description "Source IP address used by the Exporting Process.
          If not configured by the user, this parameter is set by
          the Monitoring Device to an IP address assigned to the
          outgoing interface.";
      }
      leaf destinationIPAddress {
        type inet:ip-address;
        mandatory true;
        description "IP address of the Collection Process to which
          IPFIX Messages are sent.";
      }
    }

    grouping fileWriterParameters {
      description "File Writer parameters.";
      leaf ipfixVersion {
        type uint16;
        default 10;
        description "IPFIX version number.";
        reference "RFC 5101.";
      }
      leaf file {
        type inet:uri;
        mandatory true;
        description "URI specifying the location of the file.";
      }
      leaf bytes {
        type yang:counter64;
        units octets;
        config false;
        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
          fileWriterDiscontinuityTime.";
```

```
      }
      leaf messages {
        type yang:counter64;
        units "IPFIX Messages";
        config false;
        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
          fileWriterDiscontinuityTime.";
      }
      leaf discardedMessages {
        type yang:counter64;
        units "IPFIX Messages";
        config false;
        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
          fileWriterDiscontinuityTime.";
      }
      leaf records {
        type yang:counter64;
        units "Data Records";
        config false;
        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
          fileWriterDiscontinuityTime.";
      }
      leaf templates {
        type yang:counter32;
        units "Templates";
        config false;
        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
          fileWriterDiscontinuityTime.";
      }
      leaf optionsTemplates {
        type yang:counter32;
```

```
        units "Options Templates";
        config false;
        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
          fileWriterDiscontinuityTime.";
      }
      leaf fileWriterDiscontinuityTime {
        type yang:date-and-time;
        config false;
        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.";
      }
      list template {
        config false;
        description "This list contains the Templates and Options
          Templates that have been written by the File Reader.
          Withdrawn or invalidated (Options) Templates MUST be removed
          from this list.";
        uses templateParameters;
      }
    }

    grouping optionsParameters {
      description "Parameters specifying the data export using an
        Options Template.";
      leaf optionsType {
        type identityref {
          base "optionsType";
        }
        mandatory true;
        description "Type of the exported options data.";
      }
      leaf optionsTimeout {
        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.";
      }
    }
```

```
    grouping collectingProcessParameters {
      description "Parameters of a Collecting Process.";
      list sctpCollector {
        key name;
        description "List of SCTP receivers (sockets) on which the
          Collecting Process receives IPFIX Messages.";
        leaf name {
          type nameType;
          description "Key of this list.";
        }
        uses sctpCollectorParameters;
      }
      list udpCollector {
        if-feature udpTransport;
        key name;
        description "List of UDP receivers (sockets) on which the
          Collecting Process receives IPFIX Messages.";
        leaf name {
          type nameType;
          description "Key of this list.";
        }
        uses udpCollectorParameters;
      }
      list tcpCollector {
        if-feature tcpTransport;
        key name;
        description "List of TCP receivers (sockets) on which the
          Collecting Process receives IPFIX Messages.";
        leaf name {
          type nameType;
          description "Key of this list.";
        }
        uses tcpCollectorParameters;
      }
      list fileReader {
        if-feature fileReader;
        key name;
        description "List of File Readers from which the Collecting
          Process reads IPFIX Messages.";
        leaf name {
          type nameType;
          description "Key of this list.";
        }
        uses fileReaderParameters;
      }
    }

    grouping commonCollectorParameters {
```

```
      description "Parameters of a Collecting Process that are
        common to all transport protocols.";
      leaf localPort {
        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.";
      }
      container transportLayerSecurity {
        presence "If transportLayerSecurity is present, DTLS is enabled
          if the transport protocol is SCTP or UDP, and TLS is enabled
          if the transport protocol is TCP.";
        description "TLS or DTLS configuration.";
        uses transportLayerSecurityParameters;
      }
      list transportSession {
        config false;
        description "This list contains the currently established
          Transport Sessions terminating at the given socket.";
        uses transportSessionParameters;
      }
    }

    grouping sctpCollectorParameters {
      description "Parameters of a listening SCTP socket at a
        Collecting Process.";
      uses commonCollectorParameters;
      leaf-list localIPAddress {
        type inet:ip-address;
        description "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.";
        reference "RFC 4960, Section 6.4.";
      }
    }

    grouping udpCollectorParameters {
      description "Parameters of a listening UDP socket at a
        Collecting Process.";
      uses commonCollectorParameters;
      leaf-list localIPAddress {
        type inet:ip-address;
        description "List of local IP addresses on which the Collecting
          Process listens for IPFIX Messages.";
      }
      leaf templateLifeTime {
        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 5101, the Template lifetime MUST be at
          least three times higher than the templateRefreshTimeout
          parameter value configured on the corresponding Exporting
          Processes.
          Note that this parameter corresponds to
          ipfixTransportSessionTemplateRefreshTimeout in the IPFIX
          MIB module.";
        reference "RFC 5101, Section 10.3.7; RFC 6615, Section 8
          (ipfixTransportSessionTemplateRefreshTimeout).";
      }
      leaf optionsTemplateLifeTime {
        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 5101, the Options Template lifetime MUST
          be at least three times higher than the
          optionsTemplateRefreshTimeout parameter value configured on
          the corresponding Exporting Processes.
          Note that this parameter corresponds to
          ipfixTransportSessionOptionsTemplateRefreshTimeout in the
          IPFIX MIB module.";
        reference "RFC 5101, Section 10.3.7; RFC 6615, Section 8
          (ipfixTransportSessionOptionsTemplateRefreshTimeout).";
      }
      leaf templateLifePacket {
        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 templateLifeTime.
          Note that this parameter corresponds to
          ipfixTransportSessionTemplateRefreshPacket in the IPFIX
          MIB module.";
        reference "RFC 5101, Section 10.3.7; RFC 6615, Section 8
          (ipfixTransportSessionTemplateRefreshPacket).";
```

```
      }
      leaf optionsTemplateLifePacket {
        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
          optionsTemplateLifeTime.
          Note that this parameter corresponds to
          ipfixTransportSessionOptionsTemplateRefreshPacket in the
          IPFIX MIB module.";
        reference "RFC 5101, Section 10.3.7; RFC 6615, Section 8
          (ipfixTransportSessionOptionsTemplateRefreshPacket).";
      }
    }

    grouping tcpCollectorParameters {
      description "Parameters of a listening TCP socket at a
        Collecting Process.";
      uses commonCollectorParameters;
      leaf-list localIPAddress {
        type inet:ip-address;
        description "List of local IP addresses on which the Collecting
          Process listens for IPFIX Messages.";
      }
    }

    grouping fileReaderParameters {
      description "File Reader parameters.";
      leaf file {
        type inet:uri;
        mandatory true;
        description "URI specifying the location of the file.";
      }
      leaf bytes {
        type yang:counter64;
        units octets;
        config false;
        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
          fileReaderDiscontinuityTime.";
      }
      leaf messages {
        type yang:counter64;
```

```
        units "IPFIX Messages";
        config false;
        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
          fileReaderDiscontinuityTime.";
      }
      leaf records {
        type yang:counter64;
        units "Data Records";
        config false;
        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
          fileReaderDiscontinuityTime.";
      }
      leaf templates {
        type yang:counter32;
        units "Templates";
        config false;
        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
          fileReaderDiscontinuityTime.";
      }
      leaf optionsTemplates {
        type yang:counter32;
        units "Options Templates";
        config false;
        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
          fileReaderDiscontinuityTime.";
      }
      leaf fileReaderDiscontinuityTime {
        type yang:date-and-time;
        config false;
        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.";
      }
      list template {
        config false;
        description "This list contains the Templates and Options
          Templates that have been read by the File Reader.
          Withdrawn or invalidated (Options) Template MUST be removed
          from this list.";
        uses templateParameters;
      }
    }

    grouping transportLayerSecurityParameters {
      description "TLS or DTLS parameters.";
      leaf-list localCertificationAuthorityDN {
        type string;
        description "Distinguished names of certification authorities
          whose certificates may be used to identify the local
          endpoint.";
        reference "RFC 5280.";
      }
      leaf-list localSubjectDN {
        type string;
        description "Distinguished names that may be used in the
          certificates to identify the local endpoint.";
        reference "RFC 5280.";
      }
      leaf-list localSubjectFQDN {
        type inet:domain-name;
        description "Fully qualified domain names that may be used to
          in the certificates to identify the local endpoint.";
        reference "RFC 5280.";
      }
      leaf-list remoteCertificationAuthorityDN {
        type string;
        description "Distinguished names of certification authorities
          whose certificates are accepted to authorize remote
          endpoints.";
        reference "RFC 5280.";
      }
      leaf-list remoteSubjectDN {
        type string;
        description "Distinguished names which are accepted in
          certificates to authorize remote endpoints.";
        reference "RFC 5280.";
      }
      leaf-list remoteSubjectFQDN {
        type inet:domain-name;
```

```
        description "Fully qualified domain names that are accepted in
          certificates to authorize remote endpoints.";
        reference "RFC 5280.";
      }
    }

    grouping templateParameters {
      description "State parameters of a Template used by an Exporting
        Process or received by a Collecting Process in a specific
        Transport Session.  Parameter names and semantics correspond to
        the managed objects in IPFIX-MIB";
      reference "RFC 5101; RFC 6615, Section 8 (ipfixTemplateEntry,
        ipfixTemplateDefinitionEntry, ipfixTemplateStatsEntry)";
      leaf observationDomainId {
        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 templateId {
        type uint16 {
          range "256..65535" {
            description "Valid range of Template IDs.";
            reference "RFC 5101";
          }
        }
        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 setId {
        type uint16;
        description "This number indicates the Set ID of the Template.
          Currently, there are two values defined.  The value 2 is used
          for Sets containing Template definitions.  The value 3 is
          used for Sets containing Options Template definitions.
          Note that this parameter corresponds to ipfixTemplateSetId
          in the IPFIX MIB module.";
        reference "RFC 6615, Section 8 (ipfixTemplateSetId).";
      }
      leaf accessTime {
        type yang:date-and-time;
        description "Used for Exporting Processes, this parameter
```

```
          contains the time when this (Options) Template was last
          sent to the Collector(s) 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.
          Note that this parameter corresponds to
          ipfixTemplateAccessTime in the IPFIX MIB module.";
        reference "RFC 6615, Section 8 (ipfixTemplateAccessTime).";
      }
      leaf templateDataRecords {
        type yang:counter64;
        description "The number of transmitted or 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
          templateDiscontinuityTime.
          Note that this parameter corresponds to
          ipfixTemplateDataRecords in the IPFIX MIB module.";
        reference "RFC 6615, Section 8 (ipfixTemplateDataRecords).";
      }
      leaf templateDiscontinuityTime {
        type yang:date-and-time;
        description "Timestamp of the most recent occasion at which
          the counter templateDataRecords 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 {
        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 ieId {
          type ieIdType;
          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 5101; RFC 6615, Section 8
            (ipfixTemplateDefinitionIeId).";
        }
        leaf ieLength {
          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 5101; RFC 6615, Section 8
            (ipfixTemplateDefinitionIeLength).";
        }
        leaf ieEnterpriseNumber {
          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 isFlowKey {
          when "../../setId = 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 isScope {
          when "../../setId = 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 transportSessionParameters {
    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 5101; RFC 6615, Section 8
      (ipfixTransportSessionEntry,
       ipfixTransportSessionStatsEntry).";
    leaf ipfixVersion {
      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.
        Hence, it is identical to the value of the configuration
        parameter ipfixVersion of the outer SctpExporter,
        UdpExporter, or TcpExporter node.
        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 sourceAddress {
      type inet:ip-address;
      description "The source address of the Exporter of the
        IPFIX Transport Session.
        If the transport protocol is SCTP, this is one of the
        potentially many IP addresses of the Exporter.
        Preferably, the source IP address of the path that is
        usually selected by the Exporter to send IPFIX Messages to
        the Collector SHOULD be used.
        Note that this parameter functionally corresponds to
        ipfixTransportSessionSourceAddressType and
        ipfixTransportSessionSourceAddress in the IPFIX MIB
        module.";
      reference "RFC 6615, Section 8
        (ipfixTransportSessionSourceAddressType,
        ipfixTransportSessionSourceAddress);
        RFC 4960, Section 6.4.";
    }
```

```
      leaf destinationAddress {
        type inet:ip-address;
        description "The destination address of the Collector of
          the IPFIX Transport Session.
          If the transport protocol is SCTP, this is one of the
          potentially many IP addresses of the Collector.
          Preferably, the destination IP address of the path that is
          usually selected by the Exporter to send IPFIX Messages to
          the Collector SHOULD be used.
          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 sourcePort {
        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 destinationPort {
        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 sctpAssocId {
        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";
      }
      leaf status {
        type transportSessionStatus;
        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
          transportSessionDiscontinuityTime.
          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
          transportSessionDiscontinuityTime.
          Note that this parameter corresponds to
          ipfixTransportSessionMessages in the IPFIX MIB module.";
        reference "RFC 6615, Section 8
```

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

```
          (ipfixTransportSessionTemplates).";
      }
      leaf optionsTemplates {
        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
          transportSessionDiscontinuityTime.
          Note that this parameter corresponds to
          ipfixTransportSessionOptionsTemplates in the IPFIX MIB
          module.";
        reference "RFC 6615, Section 8
          (ipfixTransportSessionOptionsTemplates).";
      }
      leaf transportSessionStartTime {
        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 transportSessionDiscontinuityTime {
        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).";
      }
      list template {
        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.";
        uses templateParameters;
      }
    }
```

```
    /*****************************************************************
    * Main container
    *****************************************************************/

    container ipfix {
      description "Top-level node of the IPFIX/PSAMP configuration
        data model.";
      list collectingProcess {
        if-feature collector;
        key name;
        description "Collecting Process of the Monitoring Device.";
        leaf name {
          type nameType;
          description "Key of this list.";
        }
        uses collectingProcessParameters;
        leaf-list exportingProcess {
          if-feature exporter;
          type leafref { path "/ipfix/exportingProcess/name"; }
          description "Export of received records without any
            modifications.  Records are processed by all Exporting
            Processes in the list.";
        }
      }

      list observationPoint {
        if-feature meter;
        key name;
        description "Observation Point of the Monitoring Device.";
        leaf name {
          type nameType;
          description "Key of this list.";
        }
        uses observationPointParameters;
        leaf-list selectionProcess {
          type leafref { path "/ipfix/selectionProcess/name"; }
          description "Selection Processes in this list process
            packets in parallel.";
        }
      }

      list selectionProcess {
        if-feature meter;
        key name;
        description "Selection Process of the Monitoring Device.";
        leaf name {
          type nameType;
          description "Key of this list.";
```

```
        }
        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 nameType;
            description "Key of this list.";
          }
          uses selectorParameters;
        }
        list selectionSequence {
          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 observationDomainId {
            type uint32;
            description "Observation Domain ID for which the
              Selection Sequence ID is assigned.";
          }
          leaf selectionSequenceId {
            type uint64;
            description "Selection Sequence ID used in the Selection
              Sequence (Statistics) Report Interpretation.";
          }
        }
        leaf cache {
          type leafref { path "/ipfix/cache/name"; }
          description "Cache that receives the output of the
            Selection Process.";
        }
      }

      list cache {
```

```
        if-feature meter;
        key name;
        description "Cache of the Monitoring Device.";
        leaf name {
          type nameType;
          description "Key of this list.";
        }
        leaf meteringProcessId {
          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 5101, Sections 4.1 and 4.2;
            IANA registry for IPFIX Entities,
            http://www.iana.org/assignments/ipfix.";
        }
        leaf dataRecords {
          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 cacheDiscontinuityTime {
          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).";
        }
        choice CacheType {
          mandatory true;
          description "Type of Cache and specific parameters.";
          container immediateCache {
            if-feature immediateCache;
            description "Flow expiration after the first packet;
              generation of Packet Records.";
            uses cacheLayoutParameters;
          }
          container timeoutCache {
            if-feature timeoutCache;
            description "Flow expiration after active and idle
              timeout; generation of Flow Records.";
            uses flowCacheParameters;
            uses cacheLayoutParameters;
          }
          container naturalCache {
            if-feature naturalCache;
            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 flowCacheParameters;
            uses cacheLayoutParameters;
          }
          container permanentCache {
            if-feature permanentCache;
            description "No flow expiration, periodical export with
              time interval exportInterval; generation of Flow
              Records.";
            uses flowCacheParameters;
            uses cacheLayoutParameters;
          }
        }
        leaf-list exportingProcess {
          if-feature exporter;
          type leafref { path "/ipfix/exportingProcess/name"; }
          description "Records are exported by all Exporting Processes
            in the list.";
        }
      }

      list exportingProcess {
        if-feature exporter;
        key name;
        description "Exporting Process of the Monitoring Device.";
```

```
        leaf name {
          type nameType;
          description "Key of this list.";
        }
        uses exportingProcessParameters;
      }
    }
  }
  <CODE ENDS>
```

## 7. Examples

This section shows example configurations conforming to the YANG module specified in Section 6.

### 7.1. PSAMP Device

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">

  <observationPoint>
    <name>OP at eth0 (ingress)</name>
    <observationDomainId>123</observationDomainId>
    <ifName>eth0</ifName>
    <direction>ingress</direction>
    <selectionProcess>Sampled UDP packets</selectionProcess>
```

```
      <selectionProcess>ICMP packets</selectionProcess>
    </observationPoint>

    <observationPoint>
      <name>OP at eth1</name>
      <observationDomainId>123</observationDomainId>
      <ifName>eth1</ifName>
      <selectionProcess>Sampled UDP packets</selectionProcess>
      <selectionProcess>ICMP packets</selectionProcess>
    </observationPoint>

    <selectionProcess>
      <name>Sampled UDP packets</name>
      <selector>
        <name>UDP filter</name>
        <filterMatch>
          <ieId>4</ieId>
          <value>17</value>
        </filterMatch>
      </selector>
      <selector>
        <name>10-out-of-100 sampler</name>
        <sampRandOutOfN>
          <size>10</size>
          <population>100</population>
        </sampRandOutOfN>
      </selector>
      <cache>PSAMP cache</cache>
    </selectionProcess>

    <selectionProcess>
      <name>ICMP packets</name>
      <selector>
        <name>ICMP filter</name>
        <filterMatch>
          <ieId>4</ieId>
          <value>1</value>
        </filterMatch>
      </selector>
      <cache>PSAMP cache</cache>
    </selectionProcess>

    <cache>
      <name>PSAMP cache</name>
      <immediateCache>
        <cacheLayout>
          <cacheField>
            <name>Field 1: ipHeaderPacketSection</name>
```

```
            <ieId>313</ieId>
            <ieLength>64</ieLength>
          </cacheField>
          <cacheField>
            <name>Field 2: observationTimeMilliseconds</name>
            <ieId>322</ieId>
          </cacheField>
        </cacheLayout>
      </immediateCache>
      <exportingProcess>The only exporter</exportingProcess>
    </cache>

    <exportingProcess>
      <name>The only exporter</name>
      <destination>
        <name>PR-SCTP collector</name>
        <sctpExporter>
          <destinationIPAddress>192.0.2.1</destinationIPAddress>
          <rateLimit>1000000</rateLimit>
          <timedReliability>500</timedReliability>
          <transportLayerSecurity>
            <remoteSubjectFQDN>coll-1.example.net</remoteSubjectFQDN>
          </transportLayerSecurity>
        </sctpExporter>
      </destination>
      <options>
        <name>Options 1</name>
        <optionsType>selectionSequence</optionsType>
        <optionsTimeout>0</optionsTimeout>
      </options>
      <options>
        <name>Options 2</name>
        <optionsType>selectionStatistics</optionsType>
        <optionsTimeout>30000</optionsTimeout>
      </options>
    </exportingProcess>

  </ipfix>
```

The above configuration results in one Template and six Options Templates. For the remainder of the example, we assume Template ID 256 for the Template and Template IDs 257 to 262 for the Options Templates. The Template is used to export the Packet Reports and has the following fields:

```
   Template ID: 256
   ipHeaderPacketSection (elementId = 313, length = 64)
   observationTimeMilliseconds (elementId = 322, length = 8)
```

Two Options Templates are used for the Selection Sequence Report Interpretations. The first one has one selectorId field and is used for the Selection Process "ICMP packets". The second one has two selectorId fields to describe the two selectors of the Selection Process "Sampled UDP packets".

```
Template ID: 257
Scope: selectionSequenceId (elementId = 301, length = 8)
observationPointId (elementId = 138, length = 4)
selectorId (elementId = 302, length = 4)

Template ID: 258
Scope: selectionSequenceId (elementId = 301, length = 8)
observationPointId (elementId = 138, length = 4)
selectorId (elementId = 302, length = 4)
selectorId (elementId = 302, length = 4)
```

Another Options Template is used to carry the Property Match Filtering Selector Report Interpretation for the Selectors "UDP filter" and "ICMP filter":

```
Template ID: 259
Scope: selectorId (elementId = 302, length = 4)
selectorAlgorithm (elementId = 304, length = 2)
protocolIdentifier (elementId = 4, length = 1)
```

Yet another Options Template is used to carry the Random n-out-of-N Sampling Selector Report Interpretation for the Selector "10-out-of-100 sampler":

```
Template ID: 260
Scope: selectorId (elementId = 302, length = 4)
selectorAlgorithm (elementId = 304, length = 2)
samplingSize (elementId = 319, length = 4)
samplingPopulation (elementId = 310, length = 4)
```

The last two Options Template are used to carry the Selection Sequence Statistics Report Interpretation for the Selection Processes, containing the statistics for one and two Selectors, respectively:

```
Template ID: 261
Scope: selectionSequenceId (elementId = 301, length = 8)
selectorIdTotalPktsObserved (elementId = 318, length = 8)
selectorIdTotalPktsSelected (elementId = 319, length = 8)
```

```
    Template ID: 262
    Scope: selectionSequenceId (elementId = 301, length = 8)
    selectorIdTotalPktsObserved (elementId = 318, length = 8)
    selectorIdTotalPktsSelected (elementId = 319, length = 8)
    selectorIdTotalPktsObserved (elementId = 318, length = 8)
    selectorIdTotalPktsSelected (elementId = 319, length = 8)
```

After a short runtime, 100 packets have been observed at the two Observation Points, including 20 UDP and 5 ICMP packets. 3 of the UDP packets are selected by the random sampler, which results in a total of 8 Packet Reports generated by the Cache. Under these circumstances, the complete configuration and state data of the PSAMP Device may look as follows:

```
<ipfix xmlns="urn:ietf:params:xml:ns:yang:ietf-ipfix-psamp">

  <observationPoint>
    <name>OP at eth0 (ingress)</name>
    <observationPointId>1</observationPointId>
    <observationDomainId>123</observationDomainId>
    <ifName>eth0</ifName>
    <direction>ingress</direction>
    <selectionProcess>Sampled UDP packets</selectionProcess>
    <selectionProcess>ICMP packets</selectionProcess>
  </observationPoint>

  <observationPoint>
    <name>OP at eth1</name>
    <observationPointId>2</observationPointId>
    <observationDomainId>123</observationDomainId>
    <ifName>eth1</ifName>
    <direction>both</direction>
    <selectionProcess>Sampled UDP packets</selectionProcess>
    <selectionProcess>ICMP packets</selectionProcess>
  </observationPoint>

  <selectionProcess>
    <name>Sampled UDP packets</name>
    <selector>
      <name>UDP filter</name>
      <filterMatch>
        <ieId>4</ieId>
        <value>17</value>
      </filterMatch>
      <packetsObserved>100</packetsObserved>
      <packetsDropped>80</packetsDropped>
      <selectorDiscontinuityTime>2010-03-15T00:00:00.00Z
          </selectorDiscontinuityTime>
```

```
      </selector>
      <selector>
        <name>10-out-of-100 sampler</name>
        <sampRandOutOfN>
          <size>10</size>
          <population>100</population>
        </sampRandOutOfN>
        <packetsObserved>20</packetsObserved>
        <packetsDropped>17</packetsDropped>
        <selectorDiscontinuityTime>2010-03-15T00:00:00.00Z
            </selectorDiscontinuityTime>
      </selector>
      <selectionSequence>
          <observationDomainId>123</observationDomainId>
          <selectionSequenceId>1</selectionSequenceId>
      </selectionSequence>
      <selectionSequence>
          <observationDomainId>123</observationDomainId>
          <selectionSequenceId>2</selectionSequenceId>
      </selectionSequence>
      <cache>PSAMP cache</cache>
    </selectionProcess>

    <selectionProcess>
      <name>ICMP packets</name>
      <selector>
        <name>ICMP filter</name>
        <filterMatch>
          <ieId>4</ieId>
          <value>1</value>
        </filterMatch>
        <packetsObserved>100</packetsObserved>
        <packetsDropped>95</packetsDropped>
        <selectorDiscontinuityTime>2010-03-15T00:00:00.00Z
            </selectorDiscontinuityTime>
      </selector>
      <selectionSequence>
          <observationDomainId>123</observationDomainId>
          <selectionSequenceId>3</selectionSequenceId>
      </selectionSequence>
      <selectionSequence>
          <observationDomainId>123</observationDomainId>
          <selectionSequenceId>4</selectionSequenceId>
      </selectionSequence>
      <cache>PSAMP cache</cache>
    </selectionProcess>

    <cache>
```

```
      <name>PSAMP cache</name>
      <meteringProcessId>1</meteringProcessId>
      <immediateCache>
        <cacheLayout>
          <cacheField>
            <name>Field 1: ipHeaderPacketSection</name>
            <ieId>313</ieId>
            <ieLength>64</ieLength>
          </cacheField>
          <cacheField>
            <name>Field 2: observationTimeMilliseconds</name>
            <ieId>322</ieId>
          </cacheField>
        </cacheLayout>
      </immediateCache>
      <dataRecords>8</dataRecords>
      <cacheDiscontinuityTime>2010-03-15T00:00:00.00Z
          </cacheDiscontinuityTime>
      <exportingProcess>The only exporter</exportingProcess>
    </cache>

    <exportingProcess>
      <name>The only exporter</name>
      <exportingProcessId>1</exportingProcessId>
      <exportMode>parallel</exportMode>
      <destination>
        <name>PR-SCTP collector</name>
        <sctpExporter>
          <ipfixVersion>10</ipfixVersion>
          <destinationIPAddress>192.0.2.1</destinationIPAddress>
          <destinationPort>4740</destinationPort>
          <sendBufferSize>32768</sendBufferSize>
          <rateLimit>1000000</rateLimit>
          <timedReliability>500</timedReliability>
          <transportLayerSecurity>
            <remoteSubjectFQDN>coll-1.example.net</remoteSubjectFQDN>
          </transportLayerSecurity>
          <transportSession>
            <ipfixVersion>10</ipfixVersion>
            <sourceAddress>192.0.2.100</sourceAddress>
            <destinationAddress>192.0.2.1</destinationAddress>
            <sourcePort>45687</sourcePort>
            <destinationPort>4740</destinationPort>
            <sctpAssocId>1</sctpAssocId>
            <status>active</status>
            <rate>230</rate>
            <bytes>978</bytes>
            <messages>3</messages>
```

```
            <records>19</records>
            <templates>1</templates>
            <optionsTemplates>6</optionsTemplates>
            <transportSessionStartTime>2010-03-15T00:00:00.50Z
                </transportSessionStartTime>
            <template>
              <observationDomainId>123</observationDomainId>
              <templateId>256</templateId>
              <setId>2</setId>
              <accessTime>2010-03-15T00:00:02.15Z</accessTime>
              <templateDataRecords>8</templateDataRecords>
              <templateDiscontinuityTime>2010-03-15T00:00:01.10Z
                  </templateDiscontinuityTime>
              <field>
                <ieId>313</ieId>
                <ieLength>64</ieLength>
                <ieEnterpriseNumber>0</ieEnterpriseNumber>
              </field>
              <field>
                <ieId>154</ieId>
                <ieLength>4</ieLength>
                <ieEnterpriseNumber>0</ieEnterpriseNumber>
              </field>
            </template>
            <template>
              <observationDomainId>123</observationDomainId>
              <templateId>257</templateId>
              <setId>3</setId>
              <accessTime>2010-03-15T00:00:02.15Z</accessTime>
              <templateDataRecords>2</templateDataRecords>
              <templateDiscontinuityTime>2010-03-15T00:00:01.10Z
                  </templateDiscontinuityTime>
              <field>
                <ieId>301</ieId>
                <ieLength>8</ieLength>
                <ieEnterpriseNumber>0</ieEnterpriseNumber>
                <isScope/>
              </field>
              <field>
                <ieId>138</ieId>
                <ieLength>4</ieLength>
                <ieEnterpriseNumber>0</ieEnterpriseNumber>
              </field>
              <field>
                <ieId>302</ieId>
                <ieLength>4</ieLength>
                <ieEnterpriseNumber>0</ieEnterpriseNumber>
              </field>
```

```
            </template>
            <template>
              <observationDomainId>123</observationDomainId>
              <templateId>258</templateId>
              <setId>3</setId>
              <accessTime>2010-03-15T00:00:02.15Z</accessTime>
              <templateDataRecords>2</templateDataRecords>
              <templateDiscontinuityTime>2010-03-15T00:00:01.10Z
                  </templateDiscontinuityTime>
              <field>
                <ieId>301</ieId>
                <ieLength>8</ieLength>
                <ieEnterpriseNumber>0</ieEnterpriseNumber>
                <isScope/>
              </field>
              <field>
                <ieId>138</ieId>
                <ieLength>4</ieLength>
                <ieEnterpriseNumber>0</ieEnterpriseNumber>
              </field>
              <field>
                <ieId>302</ieId>
                <ieLength>4</ieLength>
                <ieEnterpriseNumber>0</ieEnterpriseNumber>
              </field>
              <field>
                <ieId>302</ieId>
                <ieLength>4</ieLength>
                <ieEnterpriseNumber>0</ieEnterpriseNumber>
              </field>
            </template>
            <template>
              <observationDomainId>123</observationDomainId>
              <templateId>259</templateId>
              <setId>3</setId>
              <accessTime>2010-03-15T00:00:02.15Z</accessTime>
              <templateDataRecords>2</templateDataRecords>
              <templateDiscontinuityTime>2010-03-15T00:00:01.10Z
                  </templateDiscontinuityTime>
              <field>
                <ieId>302</ieId>
                <ieLength>4</ieLength>
                <ieEnterpriseNumber>0</ieEnterpriseNumber>
                <isScope/>
              </field>
              <field>
                <ieId>304</ieId>
                <ieLength>2</ieLength>
```

```
            <ieEnterpriseNumber>0</ieEnterpriseNumber>
          </field>
          <field>
            <ieId>4</ieId>
            <ieLength>1</ieLength>
            <ieEnterpriseNumber>0</ieEnterpriseNumber>
          </field>
        </template>
        <template>
          <observationDomainId>123</observationDomainId>
          <templateId>260</templateId>
          <setId>3</setId>
          <accessTime>2010-03-15T00:00:02.15Z</accessTime>
          <templateDataRecords>1</templateDataRecords>
          <templateDiscontinuityTime>2010-03-15T00:00:01.10Z
              </templateDiscontinuityTime>
          <field>
            <ieId>302</ieId>
            <ieLength>4</ieLength>
            <ieEnterpriseNumber>0</ieEnterpriseNumber>
            <isScope/>
          </field>
          <field>
            <ieId>304</ieId>
            <ieLength>2</ieLength>
            <ieEnterpriseNumber>0</ieEnterpriseNumber>
          </field>
          <field>
            <ieId>309</ieId>
            <ieLength>4</ieLength>
            <ieEnterpriseNumber>0</ieEnterpriseNumber>
          </field>
          <field>
            <ieId>310</ieId>
            <ieLength>4</ieLength>
            <ieEnterpriseNumber>0</ieEnterpriseNumber>
          </field>
        </template>
        <template>
          <observationDomainId>123</observationDomainId>
          <templateId>261</templateId>
          <setId>3</setId>
          <accessTime>2010-03-15T00:00:03.10Z</accessTime>
          <templateDataRecords>2</templateDataRecords>
          <templateDiscontinuityTime>2010-03-15T00:00:01.10Z
              </templateDiscontinuityTime>
          <field>
            <ieId>301</ieId>
```

```
        <ieLength>8</ieLength>
        <ieEnterpriseNumber>0</ieEnterpriseNumber>
        <isScope/>
      </field>
      <field>
        <ieId>318</ieId>
        <ieLength>8</ieLength>
        <ieEnterpriseNumber>0</ieEnterpriseNumber>
      </field>
      <field>
        <ieId>319</ieId>
        <ieLength>8</ieLength>
        <ieEnterpriseNumber>0</ieEnterpriseNumber>
      </field>
    </template>
    <template>
      <observationDomainId>123</observationDomainId>
      <templateId>262</templateId>
      <setId>3</setId>
      <accessTime>2010-03-15T00:00:03.10Z</accessTime>
      <templateDataRecords>2</templateDataRecords>
      <templateDiscontinuityTime>2010-03-15T00:00:01.10Z
          </templateDiscontinuityTime>
      <field>
        <ieId>301</ieId>
        <ieLength>8</ieLength>
        <ieEnterpriseNumber>0</ieEnterpriseNumber>
        <isScope/>
      </field>
      <field>
        <ieId>318</ieId>
        <ieLength>8</ieLength>
        <ieEnterpriseNumber>0</ieEnterpriseNumber>
      </field>
      <field>
        <ieId>319</ieId>
        <ieLength>8</ieLength>
        <ieEnterpriseNumber>0</ieEnterpriseNumber>
      </field>
      <field>
        <ieId>318</ieId>
        <ieLength>8</ieLength>
        <ieEnterpriseNumber>0</ieEnterpriseNumber>
      </field>
      <field>
        <ieId>319</ieId>
        <ieLength>8</ieLength>
        <ieEnterpriseNumber>0</ieEnterpriseNumber>
```

```
              </field>
            </template>
          </transportSession>
        </sctpExporter>
      </destination>
      <options>
        <name>Options 1</name>
        <optionsType>selectionSequence</optionsType>
        <optionsTimeout>0</optionsTimeout>
      </options>
      <options>
        <name>Options 2</name>
        <optionsType>selectionStatistics</optionsType>
        <optionsTimeout>30000</optionsTimeout>
      </options>
    </exportingProcess>

  </ipfix>
```

### 7.2. IPFIX Device

This configuration example demonstrates the shared usage of a Cache for maintaining Flow Records from two Observation Points belonging to different Observation Domains. Packets are selected using different Sampling techniques: count-based Sampling for the first Observation Point (eth0) and selection of all packets for the second Observation Point (eth1). The Exporting Process sends the Flow Records to a primary destination using SCTP. A UDP Collector is specified as secondary destination.

Exporting Process reliability statistics [RFC5101] are exported periodically every minute (60000 milliseconds). Selection Sequence Report Interpretations and Selector Report Interpretations [RFC5476] are exported once after configuring the Selection Processes. In total, two Selection Sequence Report Interpretations will be exported, one for each Selection Process.

```
<ipfix xmlns="urn:ietf:params:xml:ns:yang:ietf-ipfix-psamp">

  <observationPoint>
    <name>OP at eth0 (ingress)</name>
    <observationDomainId>123</observationDomainId>
    <ifName>eth0</ifName>
    <direction>ingress</direction>
    <selectionProcess>Count-based packet selection</selectionProcess>
  </observationPoint>

  <observationPoint>
```

```
      <name>OP at eth1</name>
      <observationDomainId>456</observationDomainId>
      <ifName>eth1</ifName>
      <selectionProcess>All packet selection</selectionProcess>
    </observationPoint>

    <selectionProcess>
      <name>Count-based packet selection</name>
      <selector>
        <name>Count-based sampler</name>
        <sampCountBased>
          <packetInterval>1</packetInterval>
          <packetSpace>99</packetSpace>
        </sampCountBased>
      </selector>
      <cache>Flow cache</cache>
    </selectionProcess>

    <selectionProcess>
      <name>All packet selection</name>
      <selector>
        <name>Select all</name>
        <selectAll/>
      </selector>
      <cache>Flow cache</cache>
    </selectionProcess>

    <cache>
      <name>Flow cache</name>
      <timeoutCache>
        <maxFlows>4096</maxFlows>
        <activeTimeout>5000</activeTimeout>
        <idleTimeout>10000</idleTimeout>
        <cacheLayout>
          <cacheField>
            <name>Field 1</name>
            <ieName>sourceIPv4Address</ieName>
            <isFlowKey/>
          </cacheField>
          <cacheField>
            <name>Field 2</name>
            <ieName>destinationIPv4Address</ieName>
            <isFlowKey/>
          </cacheField>
          <cacheField>
            <name>Field 3</name>
            <ieName>protocolIdentifier</ieName>
            <isFlowKey/>
```

```
            </cacheField>
            <cacheField>
              <name>Field 4</name>
              <ieName>sourceTransportPort</ieName>
              <isFlowKey/>
            </cacheField>
            <cacheField>
              <name>Field 5</name>
              <ieName>destinationTransportPort</ieName>
              <isFlowKey/>
            </cacheField>
            <cacheField>
              <name>Field 6</name>
              <ieName>flowStartMilliseconds</ieName>
            </cacheField>
            <cacheField>
              <name>Field 7</name>
              <ieName>flowEndSeconds</ieName>
            </cacheField>
            <cacheField>
              <name>Field 8</name>
              <ieName>octetDeltaCount</ieName>
            </cacheField>
            <cacheField>
              <name>Field 9</name>
              <ieName>packetDeltaCount</ieName>
            </cacheField>
          </cacheLayout>
        </timeoutCache>
        <exportingProcess>SCTP export with UDP backup</exportingProcess>
      </cache>

      <exportingProcess>
        <name>SCTP export with UDP backup</name>
        <exportMode>fallback</exportMode>
        <destination>
          <name>SCTP destination (primary)</name>
          <sctpExporter>
            <destinationPort>4739</destinationPort>
            <destinationIPAddress>192.0.2.1</destinationIPAddress>
          </sctpExporter>
        </destination>
        <destination>
          <name>UDP destination (secondary)</name>
          <udpExporter>
            <destinationPort>4739</destinationPort>
            <destinationIPAddress>192.0.2.2</destinationIPAddress>
            <templateRefreshTimeout>300</templateRefreshTimeout>
```

```
            <optionsTemplateRefreshTimeout>300
                </optionsTemplateRefreshTimeout>
          </udpExporter>
        </destination>
        <options>
          <name>Options 1</name>
          <optionsType>selectionSequence</optionsType>
          <optionsTimeout>0</optionsTimeout>
        </options>
        <options>
          <name>Options 2</name>
          <optionsType>exportingReliability</optionsType>
          <optionsTimeout>60000</optionsTimeout>
        </options>
      </exportingProcess>

   </ipfix>
```

### 7.3. Export of Flow Records and Packet Reports

This configuration example demonstrates the combined export of Flow Records and Packet Reports for a single Observation Point. One Selection Process applies random Sampling to the Observed Packet Stream. Its output is passed to a Cache generating Flow Records. In parallel, the Observed Packet Stream enters a second Selection Process that discards all non-ICMP packets and passes the selected packets to a second Cache for generating Packet Reports. The output of both Caches is exported to a single Collector using SCTP.

```
<ipfix xmlns="urn:ietf:params:xml:ns:yang:ietf-ipfix-psamp">

  <observationPoint>
    <name>OP at linecard 3</name>
    <observationDomainId>9876</observationDomainId>
    <ifIndex>4</ifIndex>
    <direction>ingress</direction>
    <selectionProcess>Sampling</selectionProcess>
    <selectionProcess>ICMP</selectionProcess>
  </observationPoint>

  <selectionProcess>
    <name>Sampling</name>
    <selector>
      <name>Random sampler</name>
      <sampUniProb>
        <probability>0.01</probability>
      </sampUniProb>
    </selector>
```

```
      <cache>Flow cache</cache>
    </selectionProcess>

    <selectionProcess>
      <name>ICMP</name>
      <selector>
        <name>ICMP filter</name>
        <filterMatch>
          <ieId>4</ieId>
          <value>1</value>
        </filterMatch>
      </selector>
      <cache>Packet reporting</cache>
    </selectionProcess>

    <cache>
      <name>Flow cache</name>
      <timeoutCache>
        <maxFlows>4096</maxFlows>
        <activeTimeout>5</activeTimeout>
        <idleTimeout>10</idleTimeout>
        <cacheLayout>
          <cacheField>
            <name>Field 1</name>
            <ieName>sourceIPv4Address</ieName>
            <isFlowKey/>
          </cacheField>
          <cacheField>
            <name>Field 2</name>
            <ieName>destinationIPv4Address</ieName>
            <isFlowKey/>
          </cacheField>
          <cacheField>
            <name>Field 6</name>
            <ieName>flowStartMilliseconds</ieName>
          </cacheField>
          <cacheField>
            <name>Field 7</name>
            <ieName>flowEndSeconds</ieName>
          </cacheField>
          <cacheField>
            <name>Field 8</name>
            <ieName>octetDeltaCount</ieName>
          </cacheField>
          <cacheField>
            <name>Field 9</name>
            <ieName>packetDeltaCount</ieName>
          </cacheField>
```

```
        </cacheLayout>
      </timeoutCache>
      <exportingProcess>Export</exportingProcess>
    </cache>

    <cache>
      <name>Packet reporting</name>
      <immediateCache>
        <cacheLayout>
          <cacheField>
            <name>Field 1</name>
            <ieId>313</ieId>
            <ieLength>64</ieLength>
          </cacheField>
          <cacheField>
            <name>Field 2</name>
            <ieId>154</ieId>
          </cacheField>
        </cacheLayout>
      </immediateCache>
      <exportingProcess>Export</exportingProcess>
    </cache>

    <exportingProcess>
      <name>Export</name>
      <destination>
        <name>SCTP collector</name>
        <sctpExporter>
          <destinationIPAddress>192.0.2.1</destinationIPAddress>
          <timedReliability>0</timedReliability>
        </sctpExporter>
      </destination>
      <options>
        <name>Options 1</name>
        <optionsType>selectionSequence</optionsType>
        <optionsTimeout>0</optionsTimeout>
      </options>
    </exportingProcess>

  </ipfix>
```

### 7.4. Collector and File Writer

This configuration example configures a Collector that writes the received data to a file.

```
<ipfix xmlns="urn:ietf:params:xml:ns:yang:ietf-ipfix-psamp">

  <collectingProcess>
    <name>SCTP collector</name>
    <sctpCollector>
      <name>Listening port 4739</name>
      <localPort>4739</localPort>
      <localIPAddress>192.0.2.1</localIPAddress>
    </sctpCollector>
    <exportingProcess>File writer</exportingProcess>
  </collectingProcess>

  <exportingProcess>
    <name>File writer</name>
    <destination>
      <name>Write to /tmp folder</name>
      <fileWriter>
        <file>file://tmp/collected-records.ipfix</file>
      </fileWriter>
    </destination>
  </exportingProcess>

</ipfix>
```

### 7.5. Deviations

Assume that a Monitoring Device has only two interfaces ifIndex=1 and ifIndex=2, which can be configured as Observation Points. The Observation Point ID is always identical to the ifIndex.

The following YANG module specifies these deviations.

```
module my-ipfix-psamp-deviation {
  namespace "urn:my-company:xml:ns:ietf-ipfix-psamp";
  prefix my;

  import ietf-ipfix-psamp { prefix ipfix; }

  deviation /ipfix:ipfix/ipfix:observationPoint/ipfix:entPhysicalIndex {
    deviate not-supported;
  }
  deviation /ipfix:ipfix/ipfix:observationPoint/ipfix:entPhysicalName {
    deviate not-supported;
  }
  deviation /ipfix:ipfix/ipfix:observationPoint/ipfix:ifName {
    deviate not-supported;
  }
  deviation /ipfix:ipfix/ipfix:observationPoint {
    deviate add {
      must "ipfix:ifIndex=1 or ipfix:ifIndex=2";
    }
  }
  deviation
      /ipfix:ipfix/ipfix:observationPoint/ipfix:observationPointId {
    deviate add {
      must "current()=../ipfix:ifIndex";
    }
  }
}
```

## 8. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [RFC6242].

There are a number of data nodes defined in this YANG module which 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:

/ipfix/observationPoint
: 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/selectionProcess
: 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/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/exportingProcess
: 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/collectingProcess
: 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.

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/observationPoint
: Parameters in this subtree may be sensitive because they reveal information about the Monitoring Device itself and the network infrastructure.

/ipfix/selectionProcess
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/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/exportingProcess
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/collectingProcess
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. IANA Considerations

This document registers a URI in the IETF XML registry [RFC3688]. Following the format in RFC 3688, the following registration is requested.

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

This document registers a YANG module in the YANG Module Names registry [RFC6020].

name: ietf-ipfix-psamp
namespace: urn:ietf:params:xml:ns:yang:ietf-ipfix-psamp
prefix: ipfix
reference: RFC 6728

## 10. Acknowledgements

The authors thank Martin Bjorklund, Andy Bierman, and Ladislav Lhotka for helping specify the configuration data model in YANG, as well as Atsushi Kobayashi, Andrew Johnson, Lothar Braun, and Brian Trammell for their valuable reviews of 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, March 1997.

[RFC5101] Claise, B., "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information", RFC 5101, January 2008.

[RFC5102] Quittek, J., Bryant, S., Claise, B., Aitken, P., and J. Meyer, "Information Model for IP Flow Information Export", RFC 5102, January 2008.

[RFC5103] Trammell, B. and E. Boschi, "Bidirectional Flow Export Using IP Flow Information Export (IPFIX)", RFC 5103, January 2008.

[RFC5475] Zseby, T., Molina, M., Duffield, N., Niccolini, S., and F. Raspall, "Sampling and Filtering Techniques for IP Packet Selection", RFC 5475, March 2009.

[RFC5476] Claise, B., Johnson, A., and J. Quittek, "Packet Sampling (PSAMP) Protocol Specifications", RFC 5476, March 2009.

[RFC5477] Dietz, T., Claise, B., Aitken, P., Dressler, F., and G. Carle, "Information Model for Packet Sampling Exports", RFC 5477, March 2009.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6021] Schoenwaelder, J., "Common YANG Data Types", RFC 6021, October 2010.

[UML] Object Management Group, "OMG Unified Modeling Language (OMG UML), Superstructure, V2.2", OMG formal/2009-02-02, February 2009.

[IANA-IPFIX]
IANA, "IP Flow Information Export (IPFIX) Entities", <http://www.iana.org/assignments/ipfix>.

## 11.2. Informative References

[RFC1141] Mallory, T. and A. Kullberg, "Incremental updating of the Internet checksum", RFC 1141, January 1990.

[RFC2863] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB", RFC 2863, June 2000.

[RFC3280] Housley, R., Polk, W., Ford, W., and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3280, April 2002.

[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, May 2008.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[RFC3758] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", RFC 3758, May 2004.

[RFC3871] Jones, G., "Operational Security Requirements for Large Internet Service Provider (ISP) IP Network Infrastructure", RFC 3871, September 2004.

[RFC3917] Quittek, J., Zseby, T., Claise, B., and S. Zander, "Requirements for IP Flow Information Export (IPFIX)", RFC 3917, October 2004.

[RFC4133] Bierman, A. and K. McCloghrie, "Entity MIB (Version 3)", RFC 4133, August 2005.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, June 2011.

[RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5470] Sadasivan, G., Brownlee, N., Claise, B., and J. Quittek, "Architecture for IP Flow Information Export", RFC 5470, March 2009.

[RFC5472] Zseby, T., Boschi, E., Brownlee, N., and B. Claise, "IP Flow Information Export (IPFIX) Applicability", RFC 5472, March 2009.

[RFC5473] Boschi, E., Mark, L., and B. Claise, "Reducing Redundancy in IP Flow Information Export (IPFIX) and Packet Sampling (PSAMP) Reports", RFC 5473, March 2009.

[RFC5474] Duffield, N., Chiou, D., Claise, B., Greenberg, A., Grossglauser, M., and J. Rexford, "A Framework for Packet Selection and Reporting", RFC 5474, March 2009.

[RFC5610] Boschi, E., Trammell, B., Mark, L., and T. Zseby, "Exporting Type Information for IP Flow Information Export (IPFIX) Information Elements", RFC 5610, July 2009.

[RFC5655] Trammell, B., Boschi, E., Mark, L., Zseby, T., and A. Wagner, "Specification of the IP Flow Information Export (IPFIX) File Format", RFC 5655, October 2009.

[RFC6110] Lhotka, L., "Mapping YANG to Document Schema Definition Languages and Validating NETCONF Content", RFC 6110, February 2011.

[RFC6526] Claise, B., Aitken, P., Johnson, A., and G. Muenz, "IP Flow Information Export (IPFIX) Per Stream Control Transmission Protocol (SCTP) Stream", RFC 6526, March 2012.

[RFC6615] Dietz, T., Kobayashi, A., Claise, B., and G. Muenz, "Definitions of Managed Objects for IP Flow Information Export", RFC 6615, June 2012.

[W3C.REC-xml-20081126]
Sperberg-McQueen, C., Yergeau, F., Bray, T., Paoli, J., and E. Maler, "Extensible Markup Language (XML) 1.0 (Fifth Edition)", World Wide Web Consortium Recommendation REC-xml-20081126, November 2008, <http://www.w3.org/TR/2008/REC-xml-20081126>.

[W3C.REC-xmlschema-0-20041028]
Walmsley, P. and D. Fallside, "XML Schema Part 0: Primer Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-0-20041028, October 2004, <http://www.w3.org/TR/2004/REC-xmlschema-0-20041028>.

[RFC6727] Dietz, T., Claise, B., and J. Quittek, "Definitions of Managed Objects for Packet Sampling", RFC 6727, October 2012.

[YANG-WEB]
Bjoerklund, M., "YANG WebHome", March 2011, <http://www.yang-central.org/>.

[IANA-ENTERPRISE-NUMBERS]
IANA, "Private Enterprise Numbers", <http://www.iana.org/assignments/enterprise-numbers>.

## Authors' Addresses

Gerhard Muenz
Technische Universitaet Muenchen
Department of Informatics
Chair for Network Architectures and Services (I8)
Boltzmannstr. 3
85748 Garching
Germany

EMail: muenz@net.in.tum.de
URI: http://www.net.in.tum.de/~muenz

Benoit Claise
Cisco Systems, Inc.
De Kleetlaan 6a b1
1831 Diegem
Belgium

Phone: +32 2 704 5622
EMail: bclaise@cisco.com

Paul Aitken
Cisco Systems, Inc.
96 Commercial Quay
Commercial Street
Edinburgh EH6 6LX
United Kingdom

Phone: +44 131 561 3616
EMail: paitken@cisco.com