IP Flow Information Export WG
Internet-Draft
Intended status: Standards Track
Expires: August 23, 2008

G. Muenz
University of Tuebingen
B. Claise
Cisco Systems, Inc.
February 20, 2008

Configuration Data Model for IPFIX and PSAMP
<draft-muenz-ipfix-configuration-04>

Status of this Memo

Copyright Notice

Abstract

This document specifies a data model for the configuration of metering processes, exporting processes, and collecting processes for IPFIX and PSAMP compliant monitoring devices. The configuration data model is encoded in Extensible Markup Language (XML). The structure of the data model is specified as a YANG module to ensure compatibility with the Netconf protocol. A YANG-to-XSD converter is available which allows generating an XML Schema Definition of the

data model.

Table of Contents

1. Open Issues

All open issues have been addressed.

Solved issues and answers to reviewer comments:

- SCTP timed reliability parameter configures lifetime before an IPFIX Message is "abandoned".
- Netconf compliance: ensured by using YANG instead of XSD.
- Direction attribute of interface/linecard can be on of "ingress", "egress", or "both".
- observationPointId, meteringProcessId, exportingProcessId, and selectorId have been added as optional configuration parameters, setting the values of the corresponding Information Elements. Note that monitoring device implementations are not obliged to support the configuration of these ids, but may set them dynamically. Currently not included is selectionSequenceId.
- Request for additional parameters concerning the composition of IPFIX Messages at the exporter, e.g. how long may the exporter wait until an expired record is exported? Waiting may be useful in order to fill up IPFIX Messages.
  We (the authors) decided not to add such parameters for the following reasons: 1) the composition of IPFIX Messages has not been described as configurable or managable in any other IPFIX document, and 2) today's configuration possibilities depend very much on the device or manufacturer. We propose to use device or manufacturer-dependent extensions of the configuration data model.

2. Introduction

IPFIX and PSAMP compliant monitoring devices (routers, switches, monitoring probes, mediators, collectors etc.) offer various configuration possibilities that allow adapting network monitoring to the goals and purposes of the application, e.g. accounting and charging, traffic analysis, performance monitoring, security monitoring etc. The use of a common device-independent configuration data model for IPFIX and PSAMP compliant monitoring devices facilitates network management and configuration, especially if monitoring devices of different implementers and/or manufacturers are deployed simultaneously. On the one hand, a device-independent configuration data model helps storing and managing the configuration data of monitoring devices in a consistent format. On the other hand, it can also be used for local and remote configuration of monitoring devices. However, this requires that monitoring devices natively support the configuration data model, or that a mapping between the configuration data model and the device-specific representation of configuration data is provided. An appropriate transport protocol is needed in the case of remote configuration.

The purpose of this document is the specification of a device-independent configuration data model that covers the commonly available configuration parameters of Metering Processes, Exporting Processes, and Collecting Processes. The data model is encoded in Extensible Markup Language (XML) [W3C.REC-xml-20040204]. An XML document conforming to the configuration data model contains the configuration data of one monitoring device. In order to ensure compatibility with the Netconf protocol [RFC4741], YANG [I-D.bjorklund-netconf-yang] is used as modeling language. If required, the YANG specification of the configuration data model can be converted into using XML Schema language [W3C.REC-xmlschema-0-20041028] using the pyang tool [YANG-WEB]. YANG provides mechanisms to augment the configuration data model with additional device-specific or vendor-specific parameters.

For the configuration of remote monitoring devices, an appropriate protocol is needed to transfer the XML encoded configuration data. The configuration data model is compatible with the Netconf protocol [RFC4741]. However, alternative protocols, such as the Simple Object Access Protocol (SOAP) [W3C.REC-soap12-part1-20070427], are also suitable for transferring XML data from a network management system to a monitoring device.

2.1. 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 [I-D.ietf-ipfix-architecture], per the requirements defined in [RFC3917]. This document 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]. [I-D.ietf-ipfix-mib] specifies the IPFIX Management Information Base. Finally [I-D.ietf-ipfix-as] 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.

2.2. PSAMP Documents Overview

The document "A Framework for Packet Selection and Reporting" [I-D.ietf-psamp-framework] describes the PSAMP framework for 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) supported by PSAMP are described in "Sampling and Filtering Techniques for IP Packet Selection" [I-D.ietf-psamp-sample-tech]. The PSAMP protocol [I-D.ietf-psamp-protocol] specifies the export of packet information from a PSAMP exporting process to a PSAMP collecting process. 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 [I-D.ietf-psamp-info]. Finally [I-D.ietf-psamp-mib] describes the PSAMP Management Information Base.

3. Terminology

This document adopts the terminologies used in [RFC5101] and [I-D.ietf-psamp-protocol]. As in [RFC5101], these specific terms have the first letter of a word capitalized when used in this document.

4. Structure of the Configuration Data Model

The IPFIX reference model in [I-D.ietf-ipfix-architecture] specifies the role and function of Metering Processes, Exporting Processes, and Collecting Processes. In [I-D.ietf-psamp-framework], the corresponding information is specified for the PSAMP architecture. IPFIX and PSAMP compliant monitoring device implementations usually maintain the separation of Metering Processes, Exporting Processes, and Collecting Processes (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]. The configuration data model enables the setting of commonly available configuration parameters for Metering Processes, Exporting Processes, and Collecting Processes. In addition, it allows specifying the composition of Metering Processes, Exporting Processes, and Collecting Processes within a monitoring device configuration.

The selection of commonly available configuration parameters is based on configuration issues discussed in the IPFIX and PSAMP documents [RFC3917], [RFC5101], [I-D.ietf-ipfix-architecture], [I-D.ietf-psamp-protocol], [I-D.ietf-psamp-framework], and [I-D.ietf-psamp-sample-tech]. Furthermore, the structure and content of the IPFIX MIB module [I-D.ietf-ipfix-mib] and the PSAMP MIB module [I-D.ietf-psamp-mib] were 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.

In the following, we use Unified Modeling Language (UML) class diagrams to explain the structure of the configuration data model. According to UML, different arrow types are used to distinguish two different types of relationship between UML classes: aggregation and association.

(a) Aggregation (b) Unidirectional association

Aggregation means that one class is part of the other. As an example, class B is part of class A in example (a). An association is a reference to an instance of another class. In example (b), class A contains a reference to an instance of class B. The indicated numbers define the multiplicity:

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

In UML class diagrams, all classes that occur with multiplicity greater than one in an aggregation relationship, and all classes that are referenced in associations must have a key which allows distinguishing different instances of the class. This key must be unique within the given scope. Regarding example (a), all instances of class B belonging to the same instance of class A must have keys; the scope is local to the given instance of class A. In example (b), all instance of class B must have unique keys as they can be referenced by multiple instances of class A (i.e., the scope is global). In YANG, there exists a corresponding rule which mandates the existence of a key for all elements which appear in lists [I-D.bjorklund-netconf-yang]. In the configuration data model, the key is a string parameter called "name" for all classes.

Figure 1 shows the main classes the configuration data model. The role of the classes can be briefly summarized as follows:

- The ObservationPoint class specifies an Observation Point (e.g. interface) of the monitoring device which is used for traffic monitoring. Furthermore, it configures Metering Processes that process the observed packets.
- The MeteringProcess class represents a Metering Process. A Metering Process requires a record cache which is represented by an instance of the Cache class. In order to enable the usage of the same record cache in multiple Metering Processes, the

MeteringProcess class contains only a reference to an instance of the Cache class. Note that the usage of the same cache implies that the Template defining the record format is identical for the corresponding Metering Processes. Additionally, the MeteringProcess class contains optional references to instances of the SelectionProcess class forming a Selection Sequence. Only those packets passing the sequence of Selection Processes enter the record cache. If no references to instances of the SelectionProcess class are specified, all observed packets enter the record cache.

- The SelectionProcess class contains the configuration parameters of a Selection Process, which is a Primitive Selector (i.e., sampler or filter). An instance of the SelectionProcess class can be referred from multiple Metering Processes, which allows the application of the same Selection Process in different Metering Processes.
- The Cache class contains configuration parameters of a cache which stores the records in the monitoring device. Configuration parameters of the Cache class specify the record format (Template), expiration parameters, and cache size. In addition, references to one or multiple Exporting Processes can be included. An instance of the Cache class can be referred from multiple Metering Processes, enabling the shared usage of the same record cache in different Metering Processes. As the Template is defined in the Cache class, using the same record cache implies that the record format is identical. Also, the same Exporting Processes will be used, as these are linked to the record cache.
- The ExportingProcess class contains configuration parameters of an Exporting Process. It defines the export parameters and destinations. An instance of the ExportingProcess class can be referred from multiple instances of the Cache class.

Figure 1: Main classes of the configuration data model

As can be seen in Figure 1, the MeteringProcess class defines references to instances of the SelectionProcess class and the Cache class. It acts as an envelope element specifying a series of Selection Processes, forming a Selection Sequence, and a record cache. The order in which the user specifies Selection Processes in the XML document corresponds to the order in which they are applied. Hence, by using UML associations instead of aggregation relationships, the same Selection Processes and record caches can be deployed in different Metering Processes. An example is given in Section 7.1. The MeteringProcess class itself is not instantiated, but specified as part of the ObservationPoint class. Using the same Metering Process with different Observation Points is achieved by referring to the same instances of the SelectionProcess class and the Cache class. Considering Selection Processes and Cache Parameters as instances (and not the complete Metering Process) corresponds to the common practice to implement Selection Processes and record caches as independent modules.

The Cache class refers to instances of the ExportingProcess class, which enables using the same Exporting Process for different Metering Processes.

The CollectingProcess class is depicted in Figure 2. It configures

one or multiple listening ports or input files using the Receiver class. If the monitoring device acts as a mediator or concentrator, the MeteringProcess class is specified as part of the CollectingProcess class. However, the CollectingProcess class also allows referring to instances of the ExportingProcess class to export the received records without modifications to a file or another collector.

Figure 2: CollectingProcess class

Each of the presented classes contains specific configuration parameters which are specified in the next section. 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.

5. Configuration Parameters

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

serving as keys are depicted in brackets.

5.1. ObservationPoint Class

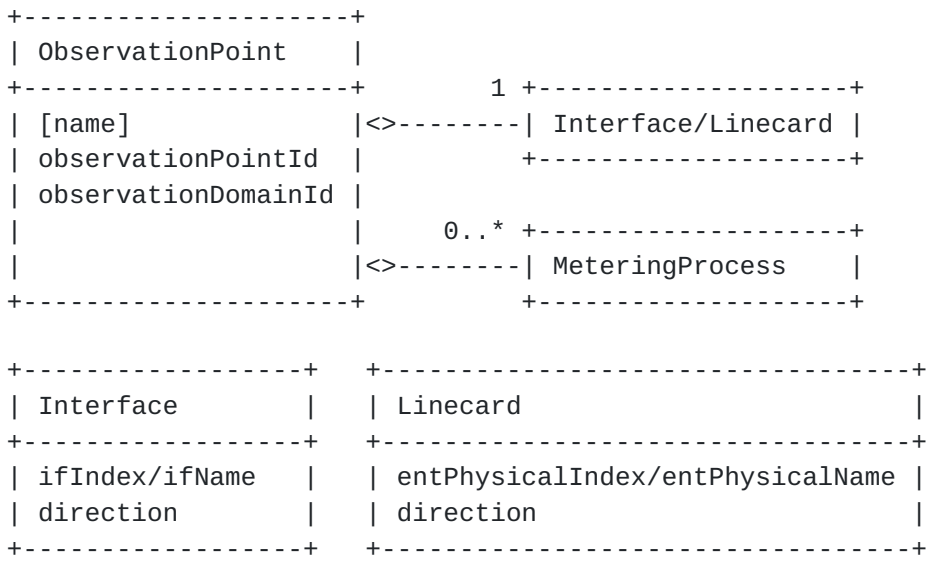

Figure 3: ObservationPoint class

The ObservationPoint class identifies an Observation Point of the monitoring device, which is either an interface or a linecard. The ObservationPoint class may specify the Observation Domain ID if the monitoring device implementation supports this configuration. If supported, the ObservationPoint class may also set the value of the Information Element observationPointId [RFC5102].

The configuration parameters to identify an interface or a linecard are as follows:

- ifIndex/ifName: Either the index or name of the interface must be specified according to corresponding objects in the IF-MIB [RFC2863].
- entPhysicalIndex/entPhysicalName: Either the index or name of the linecard must be specified according to corresponding objects in the ENTITY-MIB [RFC4133].
- direction: This parameter specifies if ingress traffic, egress traffic, or both, ingress and egress traffic is captured. If not applicable (e.g., in the case of a sniffing interface in promiscuous mode), this parameter is omitted.

The ObservationPoint class may configure one or multiple Metering Processes which process the observed packets in parallel.

5.2. MeteringProcess Class

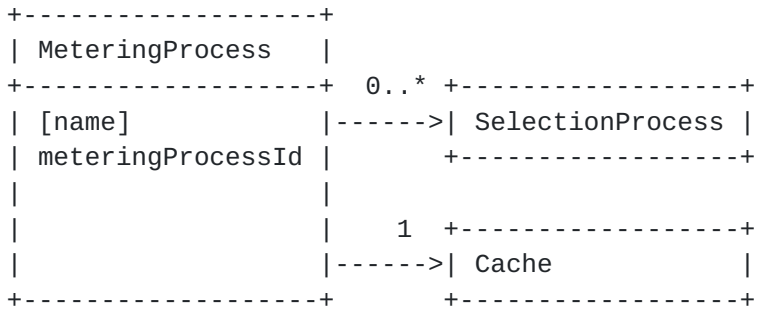

Figure 4: MeteringProcess class

The MeteringProcess class represents a Metering Process. It refers to one instance of the Cache class that specifies a record cache in the monitoring device. In addition, the MeteringProcess class may refer to one or multiple instances of the SelectionProcess class which specify sampling and filtering methods applied to the packets before entering the record cache. The order of the Selection Processes references in the XML document corresponds to the sequence in which they are applied. If no SelectionProcess is specified, all observed packets are selected. If supported by the monitoring device implementation, the MeteringProcess class may set the value of the Information Element meteringProcessId [RFC5102].

5.3. SelectionProcess Class

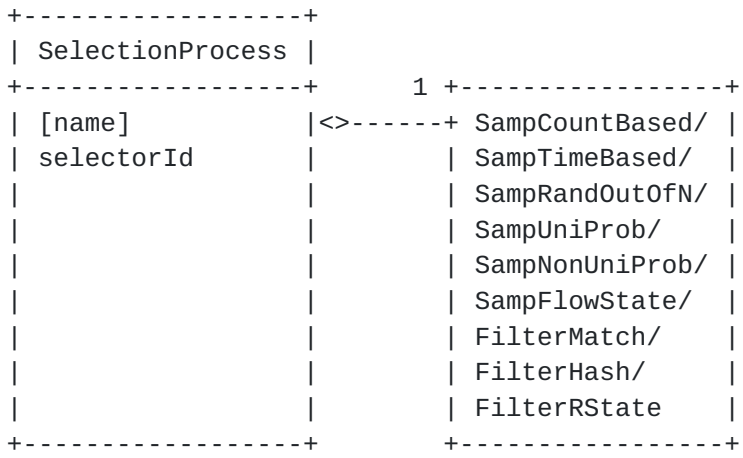

Figure 5: SelectionProcess class

The SelectionProcess class contains the configuration parameters of a Selection Process. In the configuration data model, a Selection Process implements a Primitive Selector according to [I-D.ietf-psamp-protocol]. Standardized PSAMP sampling and filtering

methods are described in [I-D.ietf-psamp-sample-tech]. The configuration parameters of each method are specified in a corresponding sampler (Samp*) or filter (Filter*) class. The SelectionProcess class contains exactly one of these classes, depending on the applied method. If supported by the monitoring device implementation, the SelectionProcess class may set the value of the Information Element selectorId [RFC5102].

5.3.1. Sampler Classes

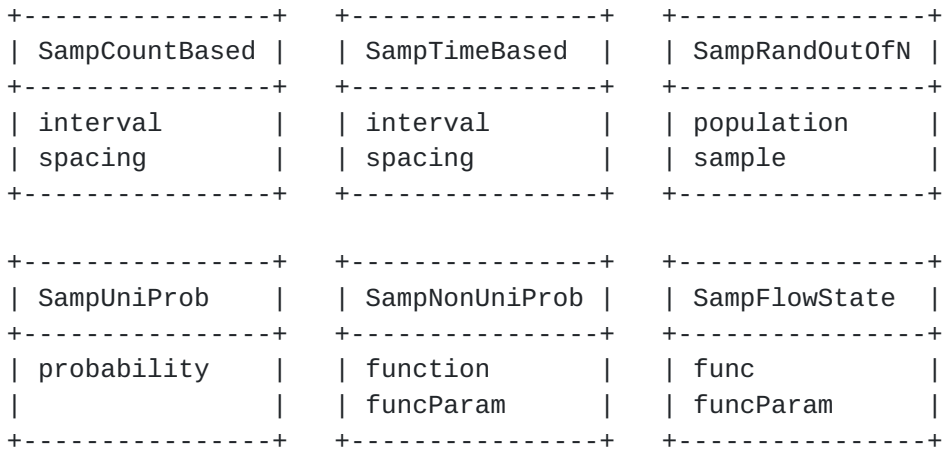

Figure 6: Sampler classes

The names and semantic of the configuration parameters correspond to the managed objects in the PSAMP MIB module [I-D.ietf-psamp-mib].

5.3.2. Filter Classes

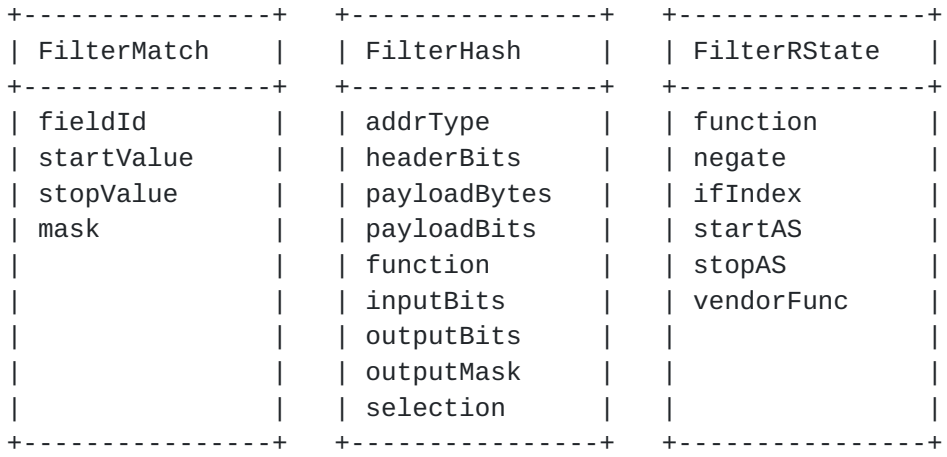

Figure 7: Filter classes

The names and semantic of the configuration parameters correspond to

the managed objects in the PSAMP MIB module [I-D.ietf-psamp-mib].

5.4. Cache Class

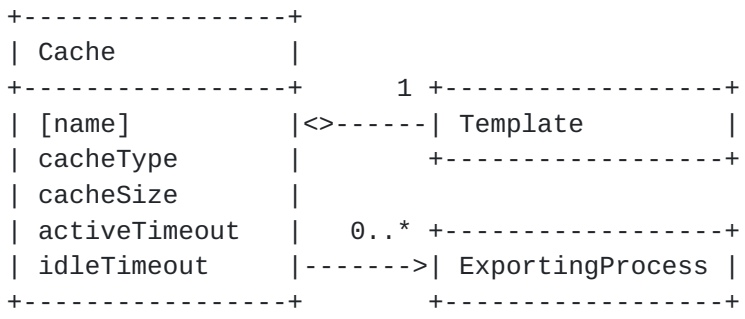

Figure 8: Cache class

The Cache class contains the configuration parameters of a record cache. The configuration parameters of the Cache class are as follows:

- cacheType: "normal", "immediate", or "permanent".
- cacheSize: maximum number of records in the cache.
- activeTimeout: timeout after which an active Flow is timed out anyway even if there is still a continuous flow of packets.
- idleTimeout: A Flow is considered to be timed out if no packets belonging to the Flow have been observed for the amount of time specified by this parameter.

The Cache class contains a Template definition which specifies the record format. Furthermore, it may refer to one or multiple instances of the ExportingProcess class, specifying the export parameters and destinations.

5.4.1. Template Class

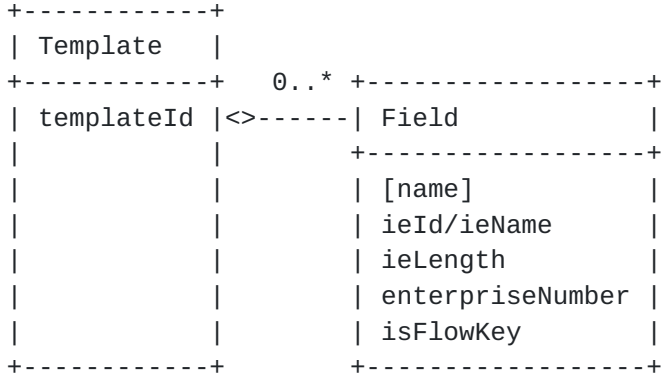

Figure 9: Template class

The Template class specifies the fields of a Template using the Field class. The configuration parameters of the Template class and the Field class are as follows:

- templateId: This is an optional parameter which allows specifying a Template ID value for the Template. As specified in the IPFIX protocol [RFC5101], the Template ID must be locally unique per Observation Domain and Transport Session, which restricts the usage of identical values for multiple Template definitions within the same monitoring device configuration. If this parameter is omitted, the Template ID will be assigned automatically by the monitoring device.
- ieId, ieName, ieLength, enterpriseNumber: These parameters specify a template field by identifier, name, length, and enterprise number of an Information Element. Either ieId or ieName must be specified. ieLength can be omitted if a default length exists of the specified Information Element. enterpriseNumber must only be inserted for enterprise-specific Information Elements.
- isFlowKey: If present, this field is a Flow Key.

The order of the fields in the XML document corresponds to the order in the Template.

5.5. ExportingProcess Class

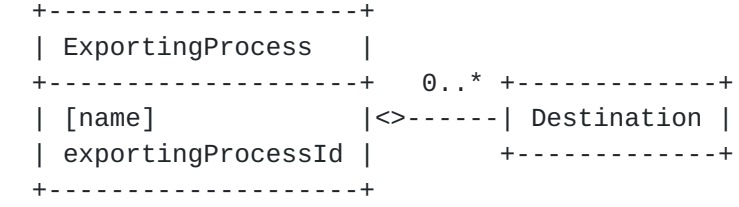

Figure 10: ExportingProcess class

The ExportingProcess class specifies a list of destinations to which the measurement data are exported. If supported by the monitoring device implementation, the ExportingProcess class may set the value of the Information Element exportingProcessId [RFC5102].

5.5.1. Destination Class

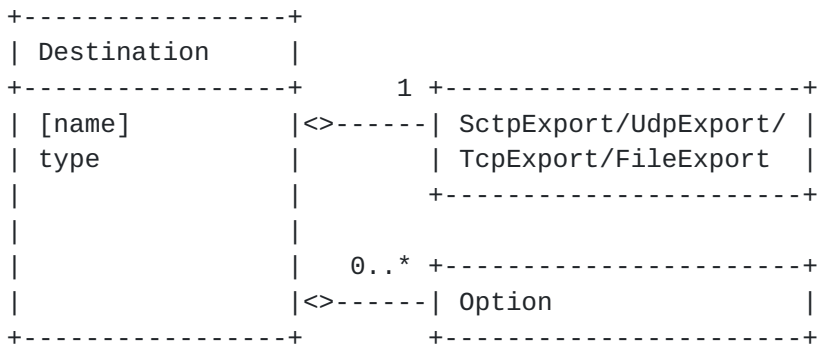

Figure 11: Destination class

The Destination class specifies one export destination of an Exporting Process. The type parameter determines the Transport Session type (primary, secondary, duplicate, load balancing, or unused) and corresponds to the ipfixTransportSessionGroupMemberType object in [I-D.ietf-ipfix-mib]. The Destination class contains further configuration parameters that are specific to the transport protocol used (SCTP, UDP, or TCP). It is also possible to export the measurement data to a file as proposed in [I-D.ietf-ipfix-file]. Optionally, the ExportingProcess class specifies the report of additional information with Option Templates, using the Option class.

5.5.2. Export Parameters Classes

Figure 12: Export parameters classes

The configuration parameters of the export parameters classes are:

- destinationIpAddress, destinationTransportPort: destination IP address and destination transport to be used for export with SCTP, UDP, or TCP.
- timedReliability: lifetime until an IPFIX Message is "abandoned" due to the timed reliability mechanism of PR-SCTP [RFC3758].
- sourceIpAddress: In the case of UdpExport, this optional parameter may appear once to set the source IP address. If this parameter is omitted, the address assigned to the outgoing interface is used.
  In the case of SctpExport, this optional parameter may appear multiple times to specify the list of eligible local IP addresses of the SCTP association [RFC4960]. If omitted, all locally assigned IP addresses are used by the SCTP endpoint.
- templateRefreshTimeout, templateRefreshPacket, optionTemplateRefreshTimeout, optionTemplateRefreshPacket: Template refresh parameters when using UDP as transport protocol.
- uri: file name and location encoded as URI if the measurement data is exported to a file.

5.5.3. Option Class

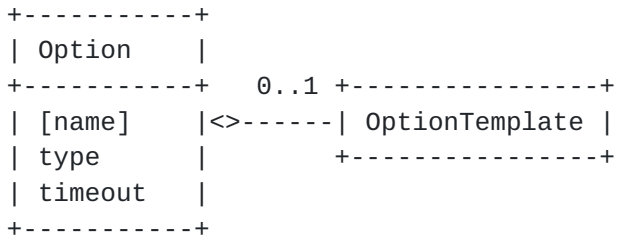

Figure 13: Option class

The Option class defines the type of additional information to be reported, such as statistics, flow keys, sampling and filtering parameters etc. [RFC5101] and [I-D.ietf-psamp-protocol] specify several types of reporting information which may be exported. The type can be one of the following:

- meteringStatistics: export of Metering Process statistics using the Metering Process Statistics Option Template [RFC5101].
- meteringReliability: export of Metering Process reliability statistics using the Metering Process Reliability Statistics Option Template [RFC5101].
- exportingReliability: export of Exporting Process reliability statistics using the Exporting Process Reliability Statistics Option Template [RFC5101].
- flowKeys: export of the Flow Key specification using the Flow Keys Option Template [RFC5101].
- selectionSequence: export of Selection Sequence and Selector Report Interpretation [I-D.ietf-psamp-protocol].
- selectionStatistics: export of Selection Sequence Statistics Report Interpretation [I-D.ietf-psamp-protocol].
- accuracy: export of Accuracy Report Interpretation [I-D.ietf-psamp-protocol].
- reducingRedundancy: export of common properties according to [I-D.ietf-ipfix-reducing-redundancy].

The Option Template can be specified manually, using the OptionTemplate class. If no Option Template is specified, the Exporter chooses a template definition automatically according to the option type and available option data.

The timeout parameter specifies the reporting interval. If the reporting timeout is zero, the corresponding reporting information will be exported only once. Otherwise, the information is exported periodically.

5.5.4. OptionTemplate Class

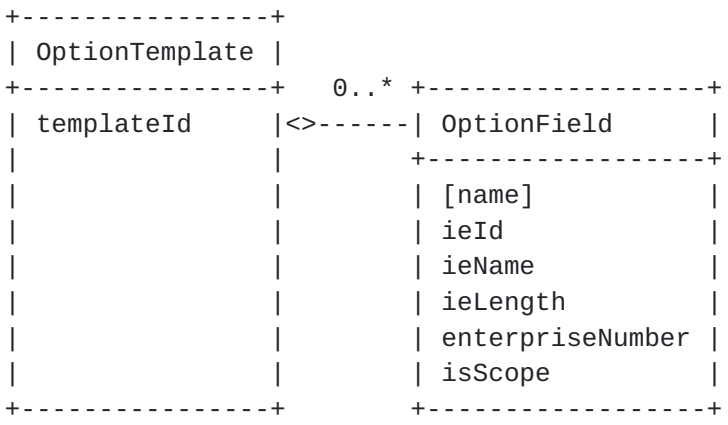

Figure 14: OptionTemplate class

The Option Template class specifies the fields of an Option Template using the OptionField class. The configuration parameters are the same as for the Template and Field classes (see Section 5.4.1). If the parameter isScope is present, the field is a scope field.

5.6. CollectingProcess Class and Receiver Class

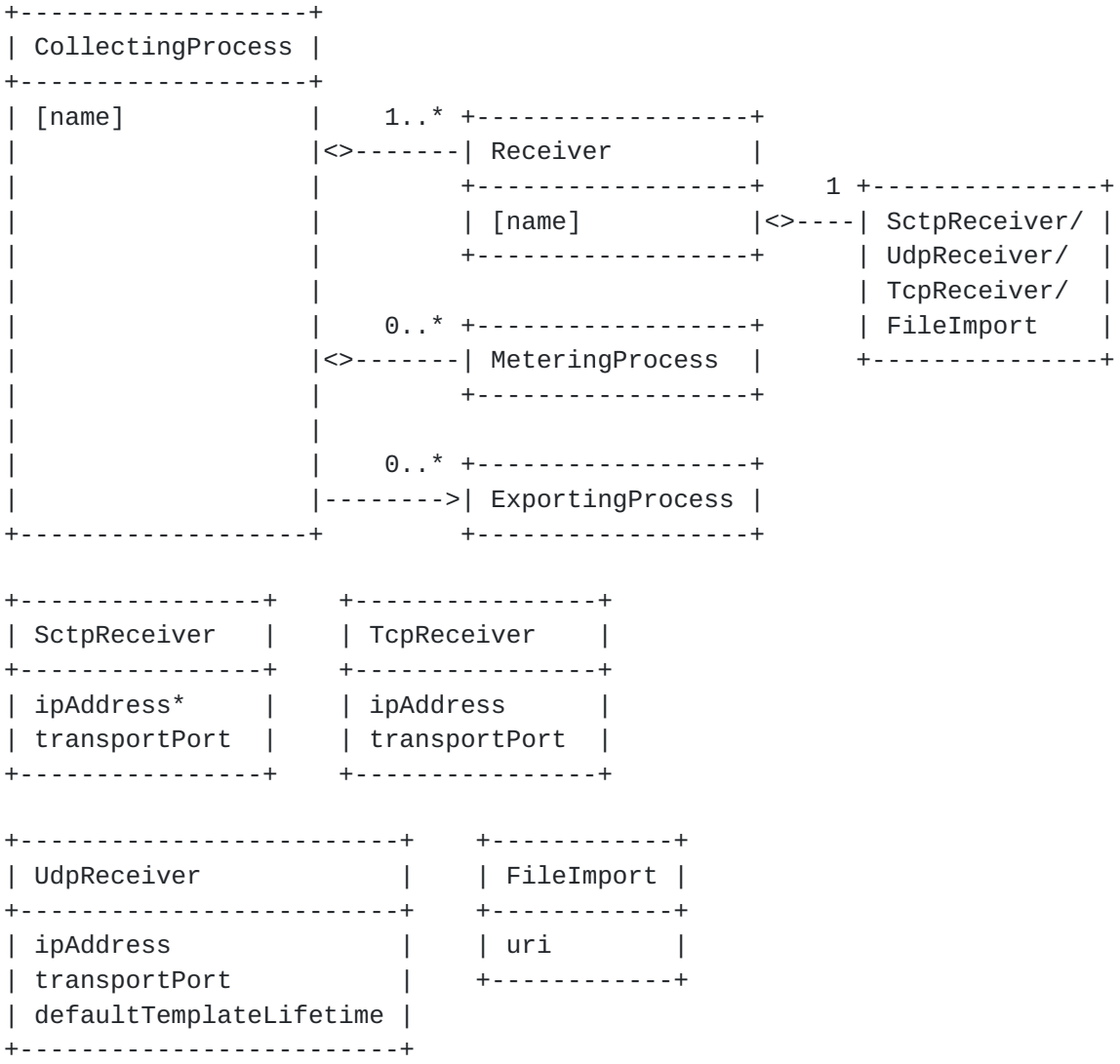

Figure 15: CollectingProcess class and Receiver Class

The CollectingProcess class contains one or multiple receivers specified with the Receiver class. The Receiver class contains further configuration parameters that are specific to the transport protocol used (SCTP, UDP, or TCP). Instead of receiving data from the network, it is possible to import it from a file to which it as been exported as proposed in [I-D.ietf-ipfix-file]. The CollectingProcess class and the SctpReceiver, UdpReceiver, TcpReceiver, and FileImport classes contain the following parameters:

- ipAddress, transportPort: IP address and port number of the receiving port. If ipAddress is omitted, the Collecting Process receives data sent to any local IP address. In the case of

SctpReceiver, multiple IP addresses can be specified as a list of eligible local IP addresses to be used for the local SCTP endpoint [RFC4960].

- o defaultTemplateLifetime: default template lifetime if UDP is used as transport protocol, ignored otherwise.
- o uri: file name and location encoded as URI if the measurement data is imported from a file.

If the monitoring device is an IPFIX mediator or concentrator as described in [I-D.kobayashi-ipfix-mediator-model] and [I-D.dressler-ipfix-aggregation], the CollectingProcess class specifies one or multiple Metering Processes.

The CollectingProcess class may refer to one or multiple instances of the ExportingProcess class in order to export received records without modifications to a file or another collector.

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

The YANG module specification of the configuration data model is specified as follows:

```
module ipfix-psamp {
  namespace "urn:ietf:params:xml:ns:ipfix-psamp-config";
  prefix ipfix;

  import yang-types { prefix yang; }
  import inet-types { prefix inet; }
  import IF-MIB { prefix if; }
  import ENTITY-MIB { prefix ent; }

  organization "IPFIX WG";
  contact "muenz@informatik.uni-tuebingen.de";

  description "IPFIX/PSAMP Configuration Data Model";

  revision 2008-02-20 {
    description "Version of draft-muenz-ipfix-configuration-04
      Changes in -04:
      - first version in yang
      - Collecting Process can be configured for file import
      - Collecting Process can be configured to export received
        records without modifications (e.g., to file or other collectors)
      - SCTP export parameter timedReliability
      - parameter for eligible local IP addresses for SCTP endpoint
      - all tags names uncapitalized, types names etc. capitalized
      - CacheParameters renamed as Cache
```

```
      - description attribute removed
      Changes in -03:
      - Linecard and Interface classes now have direction element
      - sec => s (SI unit)
      - optional description attribute for annotations
      - simplifications in ExportingProcess class
      - new parameters: observationPointId, meteringProcessId,
        selectorId, exportingProcessId (note that devices do not
        have to support the configuration of these parameters)
      - new FileExport class for exporting into a file
      - Reporting class renamed Option Class
      Changes in -02:
      - new structure without next pointers
      - packet reporting and flow metering replaced by record cache
      - added reporting with options";
  }

  grouping InformationElement {
    description "Parameters of an Information Element.";

    leaf ieEnterpriseNumber {
      description "Omitted in the case of an IETF specified Information
        Elements.";
      type uint32;
    }

    choice NameOrId {
      mandatory true;
      leaf ieName {
        type string;
      }
      leaf ieId {
        type uint16;
      }
    }

    leaf ieLength {
      description "Length can be omitted if a default length exists for
        the specified Information Element.";
      type uint16;
    }
  }


  typedef Direction {
    description "Direction of packets going through an interface or
      linecard.";
```

```
    type enumeration {
      enum ingress;
      enum egress;
      enum both;
    }
  }

  grouping Interface {
    description "Interface as input to Observation Point.";

    choice IndexOrName {
      description "Index or name of the interface as stored in the
        ifTable of IF-MIB.";
      reference "RFC 1229.";
      mandatory true;
      leaf ifIndex { type uint32; }
      leaf ifName { type string; }
    }

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

  grouping Linecard {
    description "Linecard as input to Observation Point.";

    choice IndexOrName {
      description "Index or name of the linecard as stored in the
        entPhysicalTable of ENTITY-MIB.";
      reference "RFC 4133.";
      mandatory true;
      leaf entPhysicalIndex { type uint32; }
      leaf entPhysicalName { type string; }
    }

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

  grouping MeteringProcess {
```

```
    description "Selection Processes and Record Cache of a Metering
      Process.";

    leaf meteringProcessId {
      description "If omitted, the Metering Process ID is assigned by the
        monitoring device.";
      type uint32;
    }

    leaf-list selectionProcess {
      description "Selection Processes are applied in the order of
        their appearance. If no Selection Process is specified, all
        packets are selected.";
      ordered-by user;
      type keyref { path "/ipfix/selectionProcess/name"; }
    }

    leaf cache {
      mandatory true;
      type keyref { path "/ipfix/cache/name"; }
    }
  }



  container ipfix {

    list collectingProcess {
      description "Parameters of a Collecting Process.";
      key name;

      leaf name {
        description "Arbitrary but unique name of the Collecting Process.";
        type string;
      }

      list receiver {
        description "Receiver parameters.";
        key name;

        leaf name { type string; }

        choice TransportProtocol {
          mandatory true;
          container sctpReceiver {
            description "SCTP receiver parameters.";
            reference "RFC 4960.";
            leaf-list ipAddress {
```

```
            description "List of eligible local IP addresses to be used by
              the SCTP endpoint. If omitted, all locally assigned IP
              addresses are used by the SCTP endpoint.";
            type inet:ip-address;
          }
          leaf transportPort {
            mandatory true;
            type inet:port-number;
          }
        }
        container udpReceiver {
          description "UDP receiver parameters.";
          leaf ipAddress {
            description "If omitted, all locally assigned IP addresses are
              used by the UDP endpoint.";
            type inet:ip-address;
          }
          leaf transportPort {
            mandatory true;
            type inet:port-number;
          }
          leaf defaultTemplateLifetime { type uint32; }
        }
        container tcpReceiver {
          description "TCP receiver parameters.";
          leaf ipAddress {
            description "If omitted, all locally assigned IP addresses are
              used by the TCP endpoint.";
            type inet:ip-address;
          }
          leaf transportPort {
            mandatory true;
            type inet:port-number;
          }
        }
        container fileImport {
          description "File import parameters.";
          leaf uri {
            mandatory true;
            type yang:uri;
          }
        }
      }
    }

    list meteringProcess {
      description "Metering Processes process received records in parallel.
        Monitoring device acts as IPFIX mediator/concentrator.";
```

```
        key name;

        leaf name {
          description "Arbitrary but unique name of the Monitoring
            Process.";
          type string;
        }

        uses MeteringProcess;
      }

      leaf-list exportingProcess {
        description "Export of received records without any modifications.
          Records are exported by all Exporting Processes in the list.";
        type keyref { path "/ipfix/exportingProcess/name"; }
      }
    }

    list observationPoint {
      description "Parameters of an Observation Point.";
      key name;

      leaf name {
        description "Arbitrary but unique name of the Observation Point.";
        type string;
      }

      leaf observationPointId {
        description "If omitted, the Observation Point ID is assigned by the
          monitoring device.";
        type uint32;
      }

      leaf observationDomainId {
        description "If omitted, the Observation Domain ID is assigned by the
          monitoring device.";
        type uint32;
      }

      choice OPType {
        mandatory true;
        container interface { uses Interface; }
        container linecard { uses Linecard; }
      }

      list meteringProcess {
        description "Metering Processes process packets in parallel.";
        key name;
```

```
        leaf name {
          description "Arbitrary but unique name of the Monitoring
            Process.";
          type string;
        }

        uses MeteringProcess;
      }
    }

    list selectionProcess {
      description "Parameters of a Selection Process (i.e., Primitive
        Selector).";
      key name;

      leaf name {
        description "Arbitrary but unique name of the Selection Process.";
        type string;
      }

      leaf selectorId {
        description "If omitted, the Selector ID is assigned by the
          monitoring device.";
        type uint32;
      }

      choice Method {
        description "See PSAMP-MIB for details about the selection methods
          and their parameters.";
        reference "draft-ietf-psamp-mib-06.";
        mandatory true;
        container sampCountBased {
          leaf interval { type uint32; }
          leaf spacing { type uint32; }
        }
        container sampTimeBased {
          leaf interval { type uint32; }
          leaf spacing { type uint32; }
        }
        container sampRandOutOfN {
          leaf population { type uint32; }
          leaf sample { type uint32; }
        }
        container sampUniProb {
          leaf probability {
            description "The given value must be divided by 4294967295.";
            type uint32;
          }
```

```
        }
        container sampNonUniProb {
          description "In PSAMP-MIB, these are OIDs.";
          leaf function { type string; }
          leaf funcParam { type string; }
        }
        container sampFlowState {
          description "In PSAMP-MIB, these are OIDs.";
          leaf function { type string; }
          leaf funcParam { type string; }
        }
        container filterMatch {
          leaf fieldId { type uint32; }
          leaf startValue { type string; }
          leaf stopValue { type string; }
          leaf mask { type string; }
        }
        container filterHash {
          description "In PSAMP-MIB, function and funcParam are OIDs.";
          leaf addrType { type inet:ip-version; }
          leaf headerBits { type string; }
          leaf payloadBytes { type uint32; }
          leaf payloadBits { type string; }
          leaf function { type string; }
          leaf funcParam { type string; }
          leaf inputBits { type uint32; }
          leaf outputBits { type uint32; }
          leaf outputMask { type string; }
          leaf selection { type string; }
        }
        container filterRState {
          description "In PSAMP-MIB, vendorFunc is OID.";
          leaf function { type int32; }
          leaf negate { type boolean; }
          leaf ifIndex {
            description "Index of the interface as stored in the ifTable
              of IF-MIB.";
            reference "RFC 2863.";
            type uint32;
          }
          leaf startAS { type inet:asn; }
          leaf stopAS { type inet:asn; }
          leaf vendorFunc { type string; }
        }
      }
    }

    list cache {
```

```
      description "Parameters of a cache.";
      key name;

      leaf name {
        description "Arbitrary but unique name of the cache.";
        type string;
      }

      leaf cacheType {
        type enumeration {
          enum normal {
            description "Flow expiration after active and idle timeout.";
          }
          enum immediate {
            description "Flow expiration after the first packet (PSAMP
export).";
          }
          enum permanent {
            description "No flow expiration, periodical export after active
timeout.";
          }
        }
      }

      leaf cacheSize { type uint32; }

      leaf activeTimeout { type yang:timeticks; }

      leaf idleTimeout { type yang:timeticks; }

      container template {
        leaf templateId {
          description "If omitted, the Template ID is assigned by the
            monitoring device.";
          type uint16;
        }

        list field {
          key name;
          ordered-by user;

          leaf name { type string; }

          uses InformationElement;

          leaf isFlowKey { type empty; }
        }
      }
```

```
      leaf-list exportingProcess {
```

```
        description "Records are exported by all Exporting Processes in the
list.";
        type keyref { path "/ipfix/exportingProcess/name"; }
      }
    }

    list exportingProcess {
      description "Parameters of an Exporting Process.";
      key name;

      leaf name {
        description "Arbitrary but unique name of the Exporting Process.";
        type string;
      }

      leaf exportingProcessId {
        description "If omitted, the Exporting Process ID is assigned by the
          monitoring device.";
        type uint32;
      }

      list destination {
        key name;

        leaf name { type string; }

        leaf type {
          description "Transport Session type according to IPFIX-MIB";
          reference "draft-ietf-ipfix-mib-02.";
          type enumeration {
            enum primary;
            enum secondary;
            enum duplicate;
            enum loadBalancing;
            enum unused;
          }
        }

        choice TransportProtocol {
          mandatory true;
          container sctpExport {
            description "SCTP export parameters.";
            reference "RFC 3758, RFC 4960.";
            leaf destinationIpAddress {
              mandatory true;
              type inet:ip-address;
            }
            leaf destinationTransportPort {
```

```
            mandatory true;
```

```
            type inet:port-number;
          }
          leaf-list sourceIpAddress {
            description "List of eligible local IP addresses to be used by
              the SCTP endpoint. If omitted, all locally assigned IP
              addresses are used by the local endpoint.";
            type inet:ip-address;
          }
          leaf timedReliability { type yang:timeticks; }
        }
        container udpExport {
          description "UDP export parameters.";
          leaf destinationIpAddress {
            mandatory true;
            type inet:ip-address;
          }
          leaf destinationTransportPort {
            mandatory true;
            type inet:port-number;
          }
          leaf sourceIpAddress {
            description "Source IP address. If omitted, the address
              assigned to the outgoing interface is used.";
            type inet:ip-address;
          }
          leaf templateRefreshTimeout { type yang:timeticks; }
          leaf templateRefreshPacket { type uint32; }
          leaf optionTemplateRefreshTimeout { type yang:timeticks; }
          leaf optionTemplateRefreshPacket { type uint32; }
        }
        container tcpExport {
          description "TCP export parameters.";
          leaf destinationIpAddress {
            mandatory true;
            type inet:ip-address;
          }
          leaf destinationTransportPort {
            mandatory true;
            type inet:port-number;
          }
        }
        container fileExport {
          description "File export parameters.";
          leaf uri {
            mandatory true;
            type yang:uri;
          }
        }
```

```
        }

        list option {
          key name;
          leaf name { type string; }
          leaf type {
            mandatory true;
            type enumeration {
              enum "meteringStatistics" {
                description "Metering Process Statistics.";
                reference "RFC 5101, section 4.1.";
              }
              enum "meteringReliability" {
                description "Metering Process Reliability Statistics.";
                reference "RFC 5101, section 4.2.";
              }
              enum "exportingReliability" {
                description "Exporting Process Reliability Statistics.";
                reference "RFC 5101, section 4.3.";
              }
              enum "flowKeys" {
                description "Flow Keys.";
                reference "RFC 5101, section 4.4.";
              }
              enum "selectionSequence" {
                description "Selection Sequence and Selector Reports.";
                reference "draft-ietf-psamp-protocol-09, section 6.5.1 and
                  6.5.2.";
              }
              enum "selectionStatistics" {
                description "Selection Sequence Statistics Report.";
                reference "draft-ietf-psamp-protocol-09, section 6.5.3.";
              }
              enum "accuracy" {
                description "Accuracy Report.";
                reference "draft-ietf-psamp-protocol-09, section 6.5.4.";
              }
              enum "reducingRedundancy" {
                description "Application of ipfix-reducing-redundancy.";
              }
            }
          }
          leaf timeout {
            description "Time interval for exporting option data.";
            type yang:timeticks;
          }
          container optionTemplate {
            description "If no Option Template is specified, the Exporter
```

```
              defines a template according to option type and available
              option data.";
            leaf templateId {
              description "If omitted, the Template ID is assigned by the
                monitoring device.";
              type uint16;
            }
            list optionField {
              key name;
              ordered-by user;
              leaf name { type string; }
              uses InformationElement;
              leaf isScope { type empty; }
            }
          }
        }
      }
    }
  }
}
```

7. Examples

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

7.1. PSAMP Monitoring Device

This example shows two PSAMP Metering Processes configured for the same Observation Point. The first Metering Process consists of a Selection Sequence out of two Selection Processes, a filter for UDP packets and a random sampler, the second is just an ICMP filter. The two Metering Processes deploy the same cache. The configuration assumes that the monitoring device supports the configuration of values for observationPointId, meteringProcessId, selectorId, and exportingProcessId. Exporter statistics are reported using a manually specified Option Template.

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

  <observationPoint>
    <name>OP at linecard 3</name>
    <observationPointId>1</observationPointId>
    <observationDomainId>12345</observationDomainId>
    <linecard>
      <entPhysicalIndex>3</entPhysicalIndex>
```

```
      </linecard>
      <meteringProcess>
        <name>Reports of sampled UDP packets</name>
        <meteringProcessId>1</meteringProcessId>
        <selectionProcess>UDP filter</selectionProcess>
        <selectionProcess>10-out-of-100 sampler</selectionProcess>
        <cache>PSAMP cache</cache>
      </meteringProcess>
      <meteringProcess>
        <name>Reports of ICMP packets</name>
        <meteringProcessId>2</meteringProcessId>
        <selectionProcess>ICMP filter</selectionProcess>
        <cache>PSAMP cache</cache>
      </meteringProcess>
    </observationPoint>

    <selectionProcess>
      <name>UDP filter</name>
      <selectorId>1</selectorId>
      <filterMatch>
        <fieldId>4</fieldId>
        <startValue>17</startValue>
        <stopValue>17</stopValue>
      </filterMatch>
    </selectionProcess>

    <selectionProcess>
      <name>ICMP filter</name>
      <selectorId>2</selectorId>
      <filterMatch>
        <fieldId>4</fieldId>
        <startValue>1</startValue>
        <stopValue>1</stopValue>
      </filterMatch>
    </selectionProcess>

    <selectionProcess>
      <name>10-out-of-100 sampler</name>
      <selectorId>3</selectorId>
      <sampRandOutOfN>
        <population>100</population>
        <sample>10</sample>
      </sampRandOutOfN>
    </selectionProcess>

    <cache>
      <name>PSAMP cache</name>
      <cacheType>immediate</cacheType>
```

```
      <cacheSize>512</cacheSize>
      <template>
        <field>
          <name>Field 1</name>
          <ieId>313</ieId>
          <ieLength>64</ieLength>
        </field>
        <field>
          <name>Field 2</name>
          <ieName>154</ieName>
        </field>
      </template>
      <exportingProcess>The only exporter</exportingProcess>
    </cache>

    <exportingProcess>
      <name>The only exporter</name>
      <exportingProcessId>1</exportingProcessId>
      <destination>
        <name>PR-SCTP collector</name>
        <type>primary</type>
        <sctpExport>
          <destinationIpAddress>192.0.2.1</destinationIpAddress>
          <destinationTransportPort>4739</destinationTransportPort>
          <timedReliability>200</timedReliability>
        </sctpExport>
        <option>
          <name>Option 1</name>
          <type>exportingReliability</type>
          <timeout>30000</timeout>
          <optionTemplate>
            <optionField>
              <name>Field 1</name>
              <ieName>exportingProcessId</ieName>
              <isScope/>
            </optionField>
            <optionField>
              <name>Field 2</name>
              <ieName>notSentPacketTotalCount</ieName>
            </optionField>
          </optionTemplate>
        </option>
      </destination>
    </exportingProcess>

  </ipfix>
```

7.2. IPFIX Monitoring Device

This example demonstrates the shared usage of a record cache in two different Metering Processes. Packets observed at two different Observation Points are selected using different sampling techniques. Selected packets from both Observation Points enter the same record cache. The Exporting Process sends the records to a primary destination using SCTP. A UDP Collector is specified as secondary, i.e. backup destination.

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

  <observationPoint>
    <name>OP at eth0 (ingress)</name>
    <observationDomainId>12345</observationDomainId>
    <interface>
      <ifName>eth0</ifName>
      <direction>ingress</direction>
    </interface>
    <meteringProcess>
      <name>Flows of sampled packets</name>
      <selectionProcess>Count-based sampler</selectionProcess>
      <cache>Flow cache</cache>
    </meteringProcess>
  </observationPoint>

  <observationPoint>
    <name>OP at eth1</name>
    <observationDomainId>12346</observationDomainId>
    <interface>
      <ifName>eth1</ifName>
    </interface>
    <meteringProcess>
      <name>Flows of sampled packets</name>
      <selectionProcess>Time-based sampler</selectionProcess>
      <cache>Flow Cache</cache>
    </meteringProcess>
  </observationPoint>

  <selectionProcess>
    <name>Count-based sampler</name>
    <sampCountBased>
      <interval>1</interval>
      <spacing>99</spacing>
    </sampCountBased>
  </selectionProcess>

  <selectionProcess>
```

```
    <name>Time-based sampler</name>
    <sampCountBased>
      <interval>20</interval>
      <spacing>980</spacing>
    </sampCountBased>
  </selectionProcess>

  <cache>
    <name>Flow cache</name>
    <cacheType>normal</cacheType>
    <cacheSize>4096</cacheSize>
    <activeTimeout>5</activeTimeout>
    <idleTimeout>10</idleTimeout>
    <template>
      <field>
        <name>Field 1</name>
        <ieName>sourceIPv4Address</ieName>
        <isFlowKey/>
      </field>
      <field>
        <name>Field 2</name>
        <ieName>destinationIPv4Address</ieName>
        <isFlowKey/>
      </field>
      <field>
        <name>Field 3</name>
        <ieName>transportProtocol</ieName>
        <isFlowKey/>
      </field>
      <field>
        <name>Field 4</name>
        <ieName>sourceTransportPort</ieName>
        <isFlowKey/>
      </field>
      <field>
        <name>Field 5</name>
        <ieName>destinationTransportPort</ieName>
        <isFlowKey/>
      </field>
      <field>
        <name>Field 6</name>
        <ieName>flowStartMilliSeconds</ieName>
      </field>
      <field>
        <name>Field 7</name>
        <ieName>flowEndSeconds</ieName>
      </field>
      <field>
```

```
        <name>Field 8</name>
        <ieName>octetDeltaCount</ieName>
      </field>
      <field>
        <name>Field 9</name>
        <ieName>packetDeltaCount</ieName>
      </field>
    </template>
    <exportingProcess>SCTP export with UDP backup</exportingProcess>
  </cache>

  <exportingProcess>
    <name>SCTP export with UDP backup</name>
    <destination>
      <name>SCTP destination</name>
      <type>primary</type>
      <sctpExport>
        <destinationIpAddress>192.0.2.1</destinationIpAddress>
        <destinationTransportPort>4739</destinationTransportPort>
      </sctpExport>
      <option>
        <name>Option 1</name>
        <type>selectionSequence</type>
        <timeout>0</timeout>
      </option>
      <option>
        <name>Option 2</name>
        <type>exportingReliability</type>
        <timeout>6000</timeout>
      </option>
    </destination>
    <destination>
      <name>UDP destination</name>
      <type>secondary</type>
      <udpExport>
        <destinationIpAddress>192.0.2.2</destinationIpAddress>
        <destinationTransportPort>4739</destinationTransportPort>
        <sourceIpAddress>127.0.0.1</sourceIpAddress>
        <templateRefreshTimeout>6000</templateRefreshTimeout>
        <optionTemplateRefreshTimeout>6000</optionTemplateRefreshTimeout>
      </udpExport>
      <option>
        <name>Option 1</name>
        <type>selectionSequence</type>
        <timeout>30000</timeout>
      </option>
    </destination>
  </exportingProcess>
```

```
</ipfix>
```

7.3. Collector Monitoring Device

This example configures a collector which writes the received records to a file.

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

  <collectingProcess>
    <name>SCTP collector</name>
    <receiver>
      <name>Listening port 4739</name>
      <sctpReceiver>
        <ipAddress>192.0.2.1</ipAddress>
        <transportPort>4739</transportPort>
      </sctpReceiver>
    </receiver>
    <exportingProcess>File writer</exportingProcess>
  </collectingProcess>

  <exportingProcess>
    <name>File writer</name>
    <destination>
      <name>File destination</name>
      <type>primary</type>
      <fileExport>
        <uri>file://tmp/collected-records.ipfix</uri>
      </fileExport>
    </destination>
  </exportingProcess>

</ipfix>
```

8. Security Considerations

The XML Schema Definition of the configuration data model has been conceived to enable its usage with different device implementations. In order to keep the XML Schema Definition simple and flexible, no precautions have been made to ensure that only complete and meaningful configurations can be specified. For example, most of the elements are declared optional. Furthermore, the necessary communication of device capabilities to the network management system and the corresponding limitations and adaptations of the configuration data model are not specified in this document. Hence,

the XML Schema Definition does not ensure that conforming XML documents describe configurations that are both complete and supported by a given device. Users should make sure that configuration data is validated and checked against the capabilities of the device before configuring it. If configuration data is incomplete, invalid or unsupported, it must be rejected by the device and the previous configuration should remain active. In addition, an error message should be returned specifying the reason for the error of any failed configuration attempt.

Appendix A. Acknowledgements

The authors thank Martin Bjorklund for helping specifying the configuration data model in YANG.

9. References

9.1. 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.

[I-D.ietf-psamp-protocol] Claise, B., "Packet Sampling (PSAMP) Protocol Specifications", draft-ietf-psamp-protocol-09 (work in progress), December 2007.

[I-D.ietf-psamp-info] Dietz, T., Dressler, F., Carle, G., Claise, B., and P. Aitken, "Information Model for Packet Sampling Exports", draft-ietf-psamp-info-07 (work in progress), October 2007.

[W3C.REC-xml-20040204] Bray, T., Maler, E., Yergeau, F., Sperberg-McQueen, C., and J. Paoli, "Extensible Markup Language (XML) 1.0 (Third Edition)", World Wide Web Consortium FirstEdition REC-xml-20040204, February 2004, <http://www.w3.org/TR/2004/REC-xml-20040204>.

[I-D.bjorklund-netconf-yang]
Bjorklund, M., "YANG - A data modeling language for NETCONF", draft-bjorklund-netconf-yang-02 (work in progress), February 2008.

9.2. Informative References

[W3C.REC-xmlschema-0-20041028]
Fallside, D. and P. Walmsley, "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>.

[RFC4741] Enns, R., "NETCONF Configuration Protocol", RFC 4741, December 2006.

[W3C.REC-soap12-part1-20070427]
Nielsen, H., Lafon, Y., Hadley, M., Mendelsohn, N., Moreau, J., Gudgin, M., and A. Karmarkar, "SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)", World Wide Web Consortium Recommendation REC-soap12-part1-20070427, April 2007, <http://www.w3.org/TR/2007/REC-soap12-part1-20070427>.

[I-D.ietf-ipfix-as]
Zseby, T., "IPFIX Applicability", draft-ietf-ipfix-as-12 (work in progress), July 2007.

[I-D.ietf-ipfix-architecture]
Sadasivan, G., "Architecture for IP Flow Information Export", draft-ietf-ipfix-architecture-12 (work in progress), September 2006.

[I-D.ietf-ipfix-mib]
Dietz, T., Kobayashi, A., and B. Claise, "Definitions of Managed Objects for IP Flow Information Export", draft-ietf-ipfix-mib-02 (work in progress), December 2007.

[I-D.ietf-ipfix-file]
Trammell, B., Boschi, E., Mark, L., Zseby, T., and A. Wagner, "An IPFIX-Based File Format", draft-ietf-ipfix-file-00 (work in progress), January 2008.

[I-D.ietf-ipfix-reducing-redundancy]
Boschi, E., "Reducing Redundancy in IP Flow Information Export (IPFIX) and Packet Sampling (PSAMP) Reports", draft-ietf-ipfix-reducing-redundancy-04 (work in progress), May 2007.

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

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

[I-D.dressler-ipfix-aggregation] Dressler, F., Sommer, C., Muenz, G., and A. Kobayashi, "IPFIX Flow Aggregation", draft-dressler-ipfix-aggregation-04 (work in progress), November 2007.

[I-D.kobayashi-ipfix-mediator-model] Kobayashi, A., Ishibashi, K., Tsuyoshi, K., and D. Matsubara, "Reference Model for IPFIX Mediators", draft-kobayashi-ipfix-mediator-model-01 (work in progress), November 2007.

[I-D.ietf-psamp-framework] Duffield, N., "A Framework for Packet Selection and Reporting", draft-ietf-psamp-framework-12 (work in progress), June 2007.

[I-D.ietf-psamp-mib] Dietz, T. and B. Claise, "Definitions of Managed Objects for Packet Sampling", draft-ietf-psamp-mib-06 (work in progress), June 2006.

[I-D.ietf-psamp-sample-tech] Zseby, T., "Sampling and Filtering Techniques for IP Packet Selection", draft-ietf-psamp-sample-tech-10 (work in progress), June 2007.

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

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

[YANG-WEB] Bjoerklund, M., "YANG WebHome", Homepage http://www.yang-central.org, February 2008.

Authors' Addresses

Gerhard Muenz
University of Tuebingen
Computer Networks and Internet
Sand 13
Tuebingen D-72076
DE

Phone: +49 7071 29-70534
Email: muenz@informatik.uni-tuebingen.de
URI: http://net.informatik.uni-tuebingen.de/~muenz

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

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

Full Copyright Statement

Intellectual Property

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).