

IPFIX Working Group
Internet-Draft
Intended Status: Standards Track
Expires: July, 2012

P. Aitken
Cisco Systems, Inc.
16 July 2012

Reporting Unobserved Fields in IPFIX
draft-aitken-ipfix-unobserved-fields-01

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>

This Internet-Draft will expire in July 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Abstract

The IPFIX protocol is designed to export information about observations, and lacks a method for reporting that observations are unavailable. This document discusses several methods for reporting when fields are unavailable, reviews the advantages and disadvantage of each, and recommends methods which should be used.

Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Table of Contents

1. Introduction	3
2. Terminology	4
2.1 New Terminology	4
3. Potential Methods	4
3.1 Zero-valued counters	4
3.2 Multiple Templates	5
3.3 CommonProperties	6
3.4 Default Values	7
3.4.1 Default of Zero	7
3.4.2 Default of all-ones	7
3.4.3 General default value	8
3.4.4 Field-specific default values.....	8
3.5 "observedFieldsIndicator" bitfield	9
3.6 Length of Zero	10
3.7 Size field	10
3.8 Structure data lists	11
3.8.1 Status list	11
3.8.2 Observed field list	11
3.8.3 Combined field and status list	12
4. Conclusion	12
5. New Information Elements	13
6. Security Considerations	13
7. IANA Considerations	13
8. References	14
8.1 Normative References	14
8.2 Informative References	14
9. Acknowledgements	14
10. Author's Address	14

TODO: would it be useful to define two new fields for reporting the start and end time of a period during which no observations were made?

TODO: how do these methods work with structured data?

<Aitken>

Expires April 2012

[Page 2]

1. Introduction

The IPFIX information model [RFC5102] contains a wide variety of fields [IANA-IPFIX], which are not always present in all traffic. For example, ICMP type and code. Indeed, some fields are mutually exclusive. For example, IPv4 / IPv6 address fields; UDP / TCP port numbers.

When an IPFIX Metering Process monitors a field, it only reports whenever appropriate traffic is seen. ie, whenever the Observed Traffic Stream [RFC5476] contains the field to be metered. No output is generated whenever the monitored field is not present.

The IPFIX protocol lacks a method for the Metering Process to actively express that although certain fields were being monitored, no relevant observations were made, that a particular field is not applicable, or that a value could not be calculated for a derived field. Therefore the Collecting Process cannot know whether the Metering Process was actively monitoring the field, and can only infer from the lack of export that no relevant observations were made.

Further, when a Metering Process monitors a combination of fields, some may be present while others are not. Therefore the Metering Process may observe values for some fields, though not for others.

The IPFIX Protocol [RFC5101] requires the Exporting Process to employ a number of templates in order to export these combinations, using one template for each observed combination of fields. Since the number of templates required grows exponentially with the number of field combinations, the templates can quickly become unmanageable. Indeed, just a few sets of mutually exclusive fields are sufficient to exhaust the template number space.

Clearly the IPFIX protocol [RFC5101] requires a method for the Metering Process to actively express that although fields were being monitored, no relevant observations were made.

Note that there are several cases where an observation may not be available for a given field:

1. Unavailable / Not applicable:

The monitored field was not present in, or not applicable to, the observed traffic.
eg, when RTP SSRC is requested for a TCP flow.

2. Not Calculated:

Although the required fields exist, something is preventing the calculation from occurring. For example, not enough data has been collected to calculate a TCP round-trip time.

This document discusses various potential methods for reporting such unobserved fields, reviews the advantages and disadvantages of each, and recommends suitable methods as an extension to the

IPFIX Protocol [RFC5101].

<Aitken>

Expires April 2012

[Page 3]

2. Terminology

IPFIX-specific terminology used in this document is defined in Section 2 of the IPFIX protocol specification [RFC5101]. As in [RFC5101], these IPFIX-specific terms have the first letter of a word capitalized when used in this document.

2.1 New Terminology

Unobserved Field

A field which a Metering Process is metering, but for which no traffic has been seen or no data is available.

3. Potential Methods

This section discusses and evaluates various possible methods for reporting Unobserved Fields.

3.1 Zero-valued counters

This method exports counters with a value of zero to indicate that no traffic was observed.

For example, the following data records use zero-valued counters to indicate that no traffic was observed from the specified source or sent to the specified destination:

```
. sourceIPv4Address = n.n.n.n, packetTotalCount = 0
. destinationIPv4Prefix = n.n.n.n, packetTotalCount = 0
. bgpSourceAsNumber = nnnn, octetTotalCount = 0
. destinationTransportPort = pppp, octetTotalCount = 0
. protocolIdentifier = P, octetTotalCount = 0
```

Clearly this only works for specific addresses, address prefixes, Autonomous systems, port numbers, protocols. The method is not suitable for exhaustively reporting all addresses, prefixes, Autonomous Systems, ports, or protocols from which no traffic was sent or received.

Therefore, while this is a useful method, it addresses a different problem. It does not meet the requirement of being able to report unobserved fields and is therefore not a solution.

<Aitken>

Expires April 2012

[Page 4]

3.2 Multiple Templates

The NetFlow v9 [RFC3954] and IPFIX [RFC5101] protocols are both template based. Templates express which fields are present in the exported Data Records.

When no value has been observed for a particular field, a new template is generated without that corresponding Information Element. Thus the Unobserved Field is simply omitted from the export. ie, no values are exported for unobserved fields.

While this prevents an incorrect or misleading value from being exported for the Unobserved Field, it may require a great many Templates to be created. The number may soon become unmanageable, or may exceed the Template number space - especially when the Metering Process is metering a large number of mutually-exclusive fields.

The IPFIX template number space is a little less than 16 bits (256 through 65535 = 65280 templates), so the combinations of 16 different unobserved fields will exhaust the number space.

Eg, IPv4 and IPv6 traffic require separate templates, in order to report either sourceIPv4Address and destinationIPv4Address, or sourceIPv6Address and destinationIPv6Address.

Again, udpSourcePort and udpDestinationPort are mutually exclusive with tcpSourcePort and tcpDestinationPort. Although this may be mitigated by using the generic sourceTransportPort and destinationTransportPort Information Elements together with the protocolIdentifier, not all traffic is port based. Eg, these port fields are mutually exclusive with icmpTypeCodeIPv4 and icmpTypeCodeIPv6.

A final example is the MPLS label stack. Depending how many MPLS labels are present in the Observed Traffic Stream, up to ten different templates may be required, containing the ten MPLS label stack Information Elements, mplsTopLabelStackSection through mplsLabelStackSection10.

The main issue with this method is that since no data is exported about unobserved fields, the Collecting Process cannot tell whether the field was not being observed by the Metering Process, or was being observed but no relevant traffic was seen.

Eg, if mplsLabelStackSection is not exported, the Collecting Process is unable to determine whether MPLS traffic is being actively monitored and no relevant traffic was seen, or MPLS traffic is not being monitored.

Therefore this method does not meet the requirement of being able to report unobserved fields and is therefore not a solution.

<Aitken>

Expires April 2012

[Page 5]

3.3 CommonProperties

Common Properties divides Data Records into a core part in which the fields are always observed, and additional parts according to which other fields are observed. These are exported using the method described in [RFC5473].

One template is required to express which fields are present in the core, and one Template is required for each combination of additional fields. Since multiple templates are required, the number of Templates may soon become unmanageable or may exceed the Template number space as discussed in section 3.2 above.

Although Common Properties [RFC5473] isn't specified for NetFlow v9 [RFC3954], there is no technical reason preventing this.

The main issue with this method is that again, like the Multiple Templates case above, no data is exported about Unobserved Fields - so the Collecting Process cannot tell whether the field was not being observed, or was being observed but no relevant traffic was seen.

Therefore this method does not meet the requirement of being able to report unobserved fields.

<Aitken>

Expires April 2012

[Page 6]

3.4 Default Values

With this method, a single Template specifies all the fields which the Metering Process was asked to observe, regardless of whether values were observed and are available for each of the fields. Fields for which no value was observed, or for which the value is unavailable, are exported with a default value. The default value can be of several kinds as discussed below.

Note that multiple default values are required if it's necessary to distinguish between the "not applicable" and "not required" cases. In general, both cases may be represented by a single value.

Since only one template is used, this scheme is trivial to implement and works for both NetFlow v9 [RFC3954] and IPFIX.

Specific default values are discussed in the following sections.

3.4.1 Default of Zero

All unobserved fields are exported with the value zero. This has the advantage that neither the Exporting Process nor the Collecting Process needs any extra knowledge about the field.

However, if zero is a valid value for the field, it will be impossible to distinguish an unobserved field from a real observation. Eg, when reporting the number of lost packets, `packetsLost = 0` seems to indicate that no traffic was lost, when it may be intended to indicate that there is no relevant information to report. Even IP addresses of `0.0.0.0` and `0::0` are valid representations. And zero is a valid value for `icmpTypeCodeIPv4` (indicating echo reply).

Therefore, although this method reports unobserved fields, it's not always possible to determine whether or not the field was observed. Therefore this method is not a suitable solution.

3.4.2 Default of all-ones

The "Default of all-ones" method is similar to the "Default of Zero" method discussed above, except that the all-ones value (`0xFF`, `0xFFFF`, `0xFFFFFFFF`, etc) is used for each Unobserved Field.

Such values are often, though not always, reserved and therefore may more clearly indicate whether or not the field was observed.

However, this method has the additional disadvantage that the default value for Unobserved Fields changes with the size of the field. Eg, 255 for 8-bit fields, 65535 for 16-bit fields, etc.

Additionally, field sorting is impacted without additional logic to recognise the "all-ones" value, since "unobserved" appears as the topmost value.

For this reason, and because these values are not always

reserved, this method is not a suitable solution.

<Aitken>

Expires April 2012

[Page 7]

3.4.3 General default value

This method is similar to the "Default of Zero" and "Default of all-ones" methods discussed above, except that another non-zero, non-0xFF...FF default value is used for each Unobserved Field. Eg, a repeating pattern of ASCII space (0x20), or a hex number such as "0xD15AB1ED".

However, it seems impossible to choose a value which would never appear in a real observation, both for existing Information Elements and for new Information Elements defined in future.

Eg, although "0XD15AB1EDD15AB1ED" may be quite unlikely, that's not enough to give a robust indication. Further, "0XD15A" is possible (eg, port 53594) and "0xD1" has a 1-in-256 chance of incorrectly indicating that a one-octet field was unobserved.

Therefore this method is not a suitable solution.

3.4.4 Field-specific default values

Each field is provided with a special "unobserved" value, which is outside the normal range of observed values. This value may vary from field to field.

When no value is observed for the field, it's exported with the "unobserved" value.

Eg, to report that the flow direction is not relevant to the current flow, the flowDirection Information Element could be exported with any value other than 0 (ingress) or 1 (egress).

Similarly, to report that the IP version is not relevant to the current flow, the ipVersion Information Element could be exported with a value between 10 and 15 (see [IANA-VERSION]).

Since the "unobserved" value is outwith the normal range of values for the field, it is possible to distinguish an unobserved field from a real observation.

However, both the Exporting Process and the Collecting Process need extra knowledge about each individual field.

Further, not all Information Elements have a suitable value. Eg, counter, time, and process ID Information Elements may use their entire range of bits.

Therefore this method suffers from the same problem as the other "Default Value" methods above, and is not a suitable solution.

<Aitken>

Expires April 2012

[Page 8]

3.5 "observedFieldsIndicator" bitfield

With this method, a single template contains Information Elements for all the fields which the Metering Process was asked to observe.

Additionally, the Template contains an "observedFieldsIndicator" bitfield similar to the "flowKeyIndicator" (see [IANA-IPFIX] #173), in that each bit corresponds to one Information Element in the flow record. Each bit in this field indicates whether or not a value was observed for the corresponding Information Element within the Data Record.

For each Data Record, the Collecting Process examines the "observedFieldsIndicator" bitfield to discover which fields were observed and which were unobserved within that Data Record. Unobserved fields may therefore be exported with any value, since they will be disregarded.

Since it uses only one template, this scheme is trivial to implement and works for both NetFlow v9 [RFC3954] and IPFIX.

However, mediators MUST understand the "observedFieldsIndicator" bitfield and correctly interpret it. eg, if the mediator is aggregating Data Records, it MUST pay attention to the bitfield in order to disregard data for unobserved fields. Additionally, if fields are added to or removed from the Flow Record, bits in the bitfield must be shifted accordingly. Therefore this requires changes to IPFIX record processing.

Note that if the "observedFieldsIndicator" bitfield is sent in an IPFIX Options Record, it expresses which fields are valid in that Options Data Record. It's not possible to use option scoping to report the "observedFieldsIndicator" bitfield for any other Record.

Although this method clearly reports unobserved fields, it's limited to a simple binary indication of whether or not a value was observed. It's not possible to give a reason, or to distinguish "Unavailable", "Not applicable" and "not calculated" as discussed in section 1.

Provided that the simple boolean indication is sufficient, this method provides a good solution.

This method requires a new "observedFieldsIndicator" Information Element, as specified in section 5, "New Information Elements".

<Aitken>

Expires April 2012

[Page 9]

3.6 Length of Zero

This method exports a single Template containing Information Elements for all the fields which the Metering Process was asked to observe. Fields for which no value was observed are exported using IPFIX variable-length encoding [RFC5101], with a length of zero. Therefore unobserved fields are actively indicated.

Fields which may be unobserved must be anticipated ahead of time and specified in the Template using IPFIX variable-length encoding [RFC5101].

While this method only requires a single Template, it doesn't work for NetFlow v9 export [RFC3954] since NetFlow v9 doesn't support variable-length encoding.

It also does not address the not-applicable versus not-calculated case discussed in section 3.5, which may be needed for some fields.

However, provided that the simple boolean indication is sufficient, and especially when variable-length encoding is already being used for the field, this method provides a good solution.

3.7 Size field

With this method, a single Template contains Information Elements for all the fields which the Metering Process was asked to observe.

In addition, a "size" or "count" field is added to the Template, indicating how many of the fields are valid.

Eg, the Template contains `mplsTopLabelStackSection`, `mplsLabelStackSection2`, ..., `mplsLabelStackSection10`.

Additionally, `mplsLabelStackDepth` is provided, indicating how many of the MPLS label elements are valid.

Clearly this method is field-specific. Instead, the solution should use a structured data list as discussed in section 3.8 below. Therefore this method is not recommended.

<Aitken>

Expires April 2012

[Page 10]

3.8 Structure data lists

With this method, a single template contains Information Elements for all the fields which the Metering Process was asked to observe. A list is appended to each Data Record to provide more information about the fields in the Template.

Several types of list are possible as described below.

3.8.1 Status list

A new Information Element defines the status of each field. Eg, observed, unavailable, not applicable, not calculated. One status is provided per Information Element.

A status list is encoded using a basicList as described in [RFC6313], and the list is appended to the Data Record.

This method is similar to the "observedFieldsIndicator" bitfield discussed in section 3.5, with the advantage that more information is available about each field.

However, this method suffers from the same mediator issue. ie, the list must be understood by mediators, and must be modified when a mediator adds fields to, or removes fields from, the Data Record.

With one octet per status, the Data Record is not overly burdened.

While this method offers some advantage over the "observedFieldsIndicator" bitfield discussed in section 3.5, it is not recommended.

3.8.2 Observed field list

A list of informationElementIndex, indicating which of the elements in the Template contains valid values, is appended to each Data Record.

Elements which appear in the Template but not in the list were not observed, not applicable, or could not be calculated.

Since informationElementIndex is 16 bits long, this method produces a list which is twice as long as that in section 3.8.1 above.

Since the order of the fields in the list is unimportant, when a mediator adds fields to the Data Record, the list can simply be extended. Therefore the mediator issue is mitigated to some extent.

However, it's not possible to tell why a particular Information Element is not available.

Since the "Combined" method discussed in section 3.8.3 below offers a better solution, this method is not recommended.

<Aitken>

Expires April 2012

[Page 11]

3.8.3 Combined field and status list

This method combines the status and field list from sections 3.8.1 and 3.8.2, by exporting a subTemplateList containing {informationElementIndex, status} pairs.

Therefore this method produces a list which is thrice as long as that in section 3.8.1 above. Additionally, a further template is required to encode the {informationElementIndex, status} pair.

The status makes it possible for the Collecting Process to understand why a particular Information Element is not available.

Since the order of the fields in the list is unimportant, when a mediator adds fields to the Data Record, the list can simply be extended. Therefore the mediator issue is mitigated to some extent.

This method is the most flexible and informative of all the structured data solutions. However, it incurs a penalty of an additional template to export the subTemplateList. Therefore this method is not recommended.

Not-TODO: could be implemented as two parallel basicLists, or a basicList of a new "Element+Status" IE.

4. Conclusion

Several methods of encoding "unobserved" fields have been presented. Each has pros and cons.

The only methods which satisfactorily meet the requirements are the "observedFieldsIndicator" bitfield in section 3.5, and the "Length of Zero" method in section 3.6.

These methods may be combined in order to report when no value is available for a given export field.

In the general case, the "observedFieldsIndicator" bitfield method specified in section 3.5 should be used.

However, when IPFIX variable-length encoding can be used, or is already being used, the "Length of Zero" method specified in section 3.6 may be used.

Therefore this document specifies an extension to the IPFIX protocol [RFC5101], such that a variable-length encoding with a length of zero indicates that no value was available for the corresponding Information Element.

The ability to indicate Unobserved Fields conveys an additional benefit: the ability for the Metering Process to indicate that it has begun to monitor a new flow, but does not yet have anything to export. Therefore, all the fields are unobserved.

<Aitken>

Expires April 2012

[Page 12]

5. New Information Elements

This document defines the following new IPFIX Information Elements:

observedFieldsIndicator

Description:

This bit field indicates which of the Information Elements within a Data Record have been observed. Each bit of the observedFieldsIndicator represents an Information Element in the Data Record with the n-th bit representing the n-th Information Element.

A bit set to value 1 indicates that the corresponding Information Element was observed and contains a valid value. A bit set to value 0 indicates that the corresponding Information Element was not observed and contains an invalid value. The Information Element value SHOULD be set to zero, and MUST be disregarded by the Collecting Process.

Information Elements which have no observedFieldsIndicator bit MUST contain a valid value.

If the Data Record contains more than 64 Information Elements, the corresponding Template SHOULD be designed such that all unobserved fields are among the first 64 Information Elements, because the observedFieldsIndicator only contains 64 bits. If the Data Record contains less than 64 Information Elements, then the bits in the observedFieldsIndicator for which no corresponding Information Element exists MUST have the value 0.

Abstract Data Type: unsigned64

Data Type Semantics: flags

ElementId: TBD1

6. Security Considerations

No additional security considerations are introduced in this document. The same security considerations as for the IPFIX protocol [RFC5101] apply.

7. IANA Considerations

Additional Information Elements to be allocated in the [IANA-IPFIX] registry per section 5, "New Information Elements."

<Aitken>

Expires April 2012

[Page 13]

8. References

8.1 Normative References

- [RFC5101] Claise, B., Ed., "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information", RFC 5101, January 2008.
- [RFC2119] S. Bradner, Key words for use in RFCs to Indicate Requirement Levels, BCP 14, RFC 2119, March 1997

8.2 Informative References

- [RFC5102] Quittek, J., Bryant, S., Claise, B., Aitken, P., and J. Meyer, "Information Model for IP Flow Information Export", RFC 5102, January 2008.
- [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.
- [RFC5476] Claise, B., Ed., "Packet Sampling (PSAMP) Protocol Specifications", RFC 5476, March 2009.
- [RFC3954] Claise, B., Ed., "Cisco Systems NetFlow Services Export Version 9", RFC 3954, October 2004.
- [RFC6313] Claise, B., Dhandapani, G., Aitken, P., and S. Yates, "Export of Structured Data in IP Flow Information Export (IPFIX)", RFC 6313, July 2011.
- [IANA-IPFIX] <http://www.iana.org/assignments/ipfix/ipfix.xml>
- [IANA-VERSION] <http://www.iana.org/assignments/version-numbers/version-numbers.xml>

9. Acknowledgements

Thanks to Aamer Akhter and Luca Deri for initial review and feedback, to Andrew Johnson for the lists method, and to Jan Novak, Andrew Johnson, Colin McDowall, and Dana Blair for reviewing the draft and providing feedback.

10. Author's Address

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

Phone: +44 131 561 3616

Email: paitken@cisco.com

<Aitken>

Expires April 2012

[Page 14]

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 28, 2012

A. Akhter
Cisco Systems
March 27, 2012

IPFIX Information Elements for Flow Performance Measurement
draft-akhter-opsawg-perfmon-ipfix-02.txt

Abstract

There is a need to be able to quantify and report the performance of network applications and the network service in handling user data. This performance data provides information essential in validating service level agreements, fault isolation as well as early warnings of greater problems. This document describes IPFIX Information Elements related to performance measurement of network based applications. In addition, to the performance information several non-metric information elements are also included to provide greater context to the reports. The measurements use audio/video applications as a base but are not restricted to these class of applications.

Status of this Memo

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

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

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

This Internet-Draft will expire on September 28, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	4
3. General Usage	5
3.1. Quality of Service (QoS) Monitoring	5
3.2. Service Level Agreement (SLA) Validation	5
3.3. Fault Isolation and Troubleshooting	6
4. New Information Elements	6
4.1. Transport Layer	6
4.1.1. perfPacketLoss	6
4.1.2. perfPacketExpected	7
4.1.3. perfPacketLossRate	8
4.1.4. perfPacketLossEvent	8
4.1.5. perfPacketInterArrivalJitterAvg	9
4.1.6. perfPacketInterArrivalJitterMin	9
4.1.7. perfPacketInterArrivalJitterMax	10
4.1.8. perfRoundTripNetworkDelay	10
4.2. User and Application Layer	11
4.2.1. perfSessionSetupDelay	11
4.3. Contextual Elements	12
4.3.1. mediaRTPSSRC	12
4.3.2. mediaRTPPayloadType	12
4.3.3. mediaCodec	13
5. Security Considerations	13
6. IANA Considerations	13
7. Acknowledgements	13
8. References	14
8.1. Normative References	14
8.2. Informative References	14
Author's Address	16

1. Introduction

Today's networks support a multitude of highly demanding and sensitive network applications. Network issues are readily apparent by the users of these applications due to the sensitivity of these applications to impaired network conditions. Examples of these network applications include applications making use of IP based audio, video, database transactions, virtual desktop interface (VDI), online gaming, cloud services and many more. In some cases, the impaired application translates directly to loss of revenue. In other cases, there may be regulatory or contractual service level agreements that motivate the network operator. Due to the sensitivity of these types of applications to impaired service it leaves a poor impression of the service on the user-- regardless of the actual performance of the network itself. In the case of an actual problem within the network service, monitoring the performance may yield a early indicator of a much more serious problem.

Due to the demanding and sensitive nature of these applications, network operators have tried to engineer their networks towards wringing better and differentiated performance. However, that same differentiated design prevents network operators from extrapolating observational data from one application to another, or from one set of synthetic (active test) test traffic to actual application performance. This gap highlights the importance of generic measurements as well as the reliance on user traffic measurements-- rather than synthetic tests.

Performance measurements on user data provide greater visibility not only into the quality of experience of the end users but also visibility into network health. With regards to network health, as flow performance is being measured, there will be visibility into the end to end performance which means that not only visibility into local network health, but also viability into remote network health. If these measurements are made at multiple points within the network (or between the network and end device) then there is not only identification that there might be an issue, but a span of area can be established where the issue might be. The resolution of the fault increases with the number of measurement points along the flow path.

The IP Flow Information Export Protocol (IPFIX) [RFC5101] provides new levels of flexibility in reporting from measurement points across the life cycle of a network based application. IPFIX can provide granular results in terms of flow specificity as well as time granularity. At the same time, IPFIX allows for summarization of data along different types of boundaries for operators that are unconcerned about specific sessions but about health of a service or a portion of the network. This document details the expression of

IPFIX Information Elements whose calculation is defined in an accompanying document.

As this document covers the reporting of these metrics via IPFIX, consideration is taken with mapping the metric's capabilities and context with the IPFIX information and data representation model. The guidelines outlined in [I-D.trammell-ipfix-ie-doctors] are used to ensure proper IPFIX information element definition.

There has been related work in this area such as [RFC2321], [I-D.huici-ipfix-sipfix], and [VoIP-monitor]. This document is also an attempt to generalize as well as standardize the reporting formats and measurement methodology.

2. Terminology

Terms used in this document that are defined in the Terminology section of the IPFIX Protocol [RFC5101] document are to be interpreted as defined there.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

In addition, the information element definitions use the following terms:

Name: Name of the information element per the IPFIX rules defined in Section 2.3 of [RFC5102]

Description: Short description of what the information element is trying to convey.

Observation Point: Where the measurement is meant to be performed. Either at an intermediate point (for example, a router) or end system.

Element Data Type: The IPFIX informationElementDataTypes as defined in Section 3.1 of [RFC5610]

Element Semantics: The IPFIX informationElementSemantics as defined in section Section 3.6 of [RFC5610]

Element Units: The IPFIX informationElementUnits as defined in section Section 3.7 of [RFC5610]

Element Range Begin: The IPFIX informationElementRangeBegin as defined in section Section 3.7 of [RFC5610]

Element Range End: The IPFIX informationElementRangeEnd as defined in section Section 3.7 of [RFC5610]

Element Id: The IPFIX global unique element ID as defined in Section 3.2 of [RFC5101]

Status: The status of the specification of this IPFIX Information Element.

3. General Usage

3.1. Quality of Service (QoS) Monitoring

The network operator needs to be able to gauge the end user's satisfaction with the network service. While there are many components of the satisfaction such as pricing, packaging, offering, etc., a major component of satisfaction is delivering a consistent service. The user builds trust on this consistency of the network service and is then to be able to run network applications-- which is of course the end goal. Without the ability to deliver a consistent service for end user network applications network operator will be left dealing with price sensitive disgruntled users with very low expectations (if they don't have choice of operator) or abandonment (if they have choice).

For QoS monitoring, it is important to be able to capture the application context. For example, in the case of interactive audio flows, the codec and the fact that the application is interactive should be captured. The codec type can be used to determine loss thresholds affecting end user quality and the interactive nature would suggest thresholds over one way delay. The IPFIX reporting would need to keep this information organized together for operator to be able to perform correlated analysis.

3.2. Service Level Agreement (SLA) Validation

Similar to QoS and QoE validation, there might be contractual or regulatory requirements that need to be met by the network operator. Monitoring the performance of the flows allows the application operator, network operator as well as the end user to validate of the target service is being delivered. While there is quite a diversity in the codification of network SLAs they may eventually involve some measurement of network uptime, end to end latency, end to end jitter and perhaps service response time. In the case violation of the SLA,

the start and end times, nature and network scope of the violation needs to be captured to allow for the most accurate settling of the SLA.

3.3. Fault Isolation and Troubleshooting

It has been generally easier to troubleshoot and fix problems that are binary in nature: it either works or does not work. The host is pingable or not pingable. However, the much more difficult to resolve issues that are transitory in nature, move from location to location, more complicated than simple ICMP reachability and many times unverifiable reports by the users themselves. It is these intermittent and seemingly inconsistent network impairments that performance metrics can be extremely helpful with. Just the basic timely detection that there is a problem (or an impending problem) can give the provider the confidence that there is a real problem that needs to be resolved. The next step would be to assist the operator in a speedy resolution by providing information regarding the network location and nature of the problem.

4. New Information Elements

The information elements are organized into two main groups:

Transport Layer: Metrics that might be calculated from observations at higher layers but essentially provide information about the network transport of user data. For example, the metrics related to packet loss, latency and jitter would be defined here.

User and Application Layer: Metrics that are might be affected by the network indirectly, but are ultimately related to user, end-system and session states. For example, session setup time, transaction rate and session duration would be defined here.

Contextual Elements Information elements that provide further context to the metrics. For example, media type, codec type, and type of application would be defined here.

4.1. Transport Layer

4.1.1. perfPacketLoss

Name: perfPacketLoss

Description: The packet loss metric reports the number of individual packets that were lost in the reporting interval.

Observation Point: The observation can be made anywhere along the media path or on the endpoints them selves. The observation is only relevant in a unidirectional sense.

Element Data Type: unsigned32

Element Semantics: deltaCounter

Element Units: packets

Element Range Begin: 0

Element Range End: 0xFFFFFFFF

Element Id: TBDperfPacketLoss

Status: current

4.1.2. perfPacketExpected

Name: perfPacketExpected

Description: The number of packets there were expected within a monitoring interval.

Observation Point: The observation can be made anywhere along the media path or on the endpoints them selves. The observation is only relevant in a unidirectional sense.

Element Data Type: unsigned32

Element Semantics: deltaCounter

Element Units: none

Element Range Begin: 0

Element Range End: 0xFFFFFFFF

Element Id: TBDperfPacketExpected

Status: current

4.1.3. perfPacketLossRate

Name: perfPacketLossRate

Description: Percentage of number of packets lost out of the total set of packets sent.

Observation Point: The observation can be made anywhere along the media path or on the endpoints them selves. The observation is only relevant in a unidirectional sense.

Element Data Type: unsigned16

Element Semantics: quantity

Element Units: none

Element Range Begin: 0

Element Range End: 0xFFFE

Element Id: TBDperfPacketLossRate

Status: current

4.1.4. perfPacketLossEvent

Name: perfPacketLossEvent

Description: The packet loss event metric reports the number of continuous sets of packets that were lost in the reporting interval.

Observation Point: The observation can be made anywhere along the media path or on the endpoints them selves. The observation is only relevant in a unidirectional sense.

Element Data Type: unsigned32

Element Semantics: deltaCounter

Element Units: none

Element Range Begin: 0

Element Range End: 0xFFFFFFFF

Element Id: TBDperfPacketExpected

Status: current

4.1.5. perfPacketInterArrivalJitterAvg

Name: perfPacketInterArrivalJitterAvg

Description: This metric measures the absolute deviation of the difference in packet spacing at the measurement point compared to the packet spacing at the sender.

Observation Point: The observation can be made anywhere along the media path or on the receiver. The observation is only relevant in a unidirectional sense.

Element Data Type: unsigned32

Element Semantics: quantity

Element Units: microseconds

Element Range Begin: 0

Element Range End: 0xFFFFFFFF

Element Id: TBDperfPacketInterArrivalJitterAvg

Status: current

4.1.6. perfPacketInterArrivalJitterMin

Name: perfPacketInterArrivalJitterMin

Description: This metric measures the minimum value the calculation used for perfPacketInterArrivalJitterAvg within the monitoring interval.

Observation Point: The observation can be made anywhere along the media path or on the receiver. The observation is only relevant in a unidirectional sense.

Element Data Type: unsigned32

Element Semantics: quantity
Element Units: microseconds
Element Range Begin: 0
Element Range End: 0xFFFFFFFFE
Element Id: TBDperfPacketInterArrivalJitterMin
Status: current

4.1.7. perfPacketInterArrivalJitterMax

Name: perfPacketInterArrivalJitterMax
Description: This metric measures the maximum value the calculation used for perfPacketInterArrivalJitterAvg within the monitoring interval.
Observation Point: The observation can be made anywhere along the media path or on the receiver. The observation is only relevant in a unidirectional sense.
Element Data Type: unsigned32
Element Semantics: quantity
Element Units: microseconds
Element Range Begin: 0
Element Range End: 0xFFFFFFFFE
Element Id: TBDperfPacketInterArrivalJitterMax
Status: current

4.1.8. perfRoundTripNetworkDelay

Name: perfRoundTripNetworkDelay
Description: This metric measures the network round trip time between end stations for a flow.

Observation Point: The observation can be made anywhere along the flow path as long as the bidirectional network delay is accounted for.

Element Data Type: unsigned32

Element Semantics: quantity

Element Units: microseconds

Element Range Begin: 0

Element Range End: 0xFFFFFFFF

Element Id: TBDperfRoundTripNetworkDelay

Status: current

4.2. User and Application Layer

4.2.1. perfSessionSetupDelay

Name: perfSessionSetupDelay

Description: The Session Setup Delay metric reports the time taken from a request being initiated by a host/endpoint to the response (or request indicator) to the request being observed. This metric is defined in [RFC4710], however the units have been updated to microseconds.

Observation Point: This metric needs to be calculated where both request and response can be observed. This could be at network choke points, application proxies, or within the end systems themselves.

Element Data Type: unsigned32

Element Semantics: quantity

Element Units: microseconds

Element Range Begin: 0

Element Range End: 0xFFFFFFFF

Element Id: TBDperfSessionSetupDelay

Status: current

4.3. Contextual Elements

4.3.1. mediaRTPSSRC

Name: mediaRTPSSRC

Description: Value of the synchronization source (SSRC) field in the RTP header of the flow. This field is defined in [RFC3550]

Observation Point: This metric can be gleaned from the RTP packets directly, so the observation point needs to be on the flow path or within the endpoints.

Element Data Type: unsigned32

Element Semantics: identifier

Element Units: octets

Element Range Begin: 0

Element Range End: 0xFFFFFFFF

Element Id: TBDmediaRTPSSRC

Status: current

4.3.2. mediaRTPPayloadType

Name: mediaRTPPayloadType

Description: The value of the RTP Payload Type Field as seen in the RTP header of the flow. This field is defined in [RFC3550]

Observation Point: This metric can be gleaned from the RTP packets directly, so the observation point needs to be on the flow path or within the endpoints.

Element Data Type: unsigned8

Element Semantics: identifier

Element Units: octets

Element Range Begin: 0

Element Range End: 0xFF

Element Id: TBDmediaRTTPayloadType

Status: current

4.3.3. mediaCodec

Name: mediaCodec

Description: The media codec used in the flow.

Observation Point: The ideal location of this metric is on the media generators and consumers. However, given application inspection or static configuration it is possible that intermediate nodes are able to generate codec information.

Element Data Type: string

Element Semantics: identifier

Element Units: octets

Element Id: TBDmediaCodec

Status: current

5. Security Considerations

The recommendations in this document do not introduce any additional security issues to those already mentioned in [RFC5101] and [RFC5477]

6. IANA Considerations

This document requires an elements assignment to be made by IANA.

7. Acknowledgements

The authors would like to thank Shingo Kashima for their invaluable review and comments. Thank-you.

8. References

8.1. Normative References

- [RFC5101] Claise, B., "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information", RFC 5101, January 2008.
- [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.
- [RFC4710] Siddiqui, A., Romascanu, D., and E. Golovinsky, "Real-time Application Quality-of-Service Monitoring (RAQMON) Framework", RFC 4710, October 2006.
- [RFC5102] Quittek, J., Bryant, S., Claise, B., Aitken, P., and J. Meyer, "Information Model for IP Flow Information Export", RFC 5102, January 2008.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC3497] Gharai, L., Perkins, C., Goncher, G., and A. Mankin, "RTP Payload Format for Society of Motion Picture and Television Engineers (SMPTE) 292M Video", RFC 3497, March 2003.
- [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.
- [I-D.ietf-pmol-sip-perf-metrics] Malas, D. and A. Morton, "Basic Telephony SIP End-to-End Performance Metrics", draft-ietf-pmol-sip-perf-metrics-07 (work in progress), September 2010.
- [iana-ipfix-assignments] Internet Assigned Numbers Authority, "IP Flow Information Export Information Elements (<http://www.iana.org/assignments/ipfix/ipfix.xml>)".

8.2. Informative References

- [I-D.ietf-pmol-metrics-framework] Clark, A. and B. Claise, "Guidelines for Considering New Performance Metric Development",

draft-ietf-pmol-metrics-framework-12 (work in progress),
July 2011.

[I-D.trammell-ipfix-ie-doctors]

Trammell, B. and B. Claise, "Guidelines for Authors and Reviewers of IPFIX Information Elements",
draft-trammell-ipfix-ie-doctors-03 (work in progress),
October 2011.

[RFC2508] Casner, S. and V. Jacobson, "Compressing IP/UDP/RTP Headers for Low-Speed Serial Links", RFC 2508,
February 1999.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.

[RFC2250] Hoffman, D., Fernando, G., Goyal, V., and M. Civanlar, "RTP Payload Format for MPEG1/MPEG2 Video", RFC 2250,
January 1998.

[RFC2890] Dommety, G., "Key and Sequence Number Extensions to GRE", RFC 2890, September 2000.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005.

[RFC5761] Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", RFC 5761, April 2010.

[I-D.huici-ipfix-sipfix]

Huici, F., Niccolini, S., and S. Anderson, "SIPFIX: Use Cases and Problem Statement for VoIP Monitoring and Exporting", draft-huici-ipfix-sipfix-00 (work in progress), June 2009.

[nProbe] "nProbe - NetFlow/IPFIX Network Probe
(<http://www.ntop.org/nProbe.html>)".

[RFC2321] Bressen, A., "RITA -- The Reliable Internetwork Troubleshooting Agent", RFC 2321, April 1998.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

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

[VoIP-monitor]

L. Chang-Yong, H. Kim, K. Ko, J. Jim, and H. Jeong, "A
VoIP Traffic Monitoring System based on NetFlow v9,
International Journal of Advanced Science and Technology,
vol. 4, Mar. 2009".

Author's Address

Aamer Akhter
Cisco Systems, Inc.
7025 Kit Creek Road
RTP, NC 27709
USA

Email: aakhter@cisco.com

IPFIX Working Group
Internet-Draft
Intended Status: Informational
Expires: January 8, 2013

B. Claise
P. Aitken
N. Ben-Dvora
Cisco Systems, Inc.
July 9, 2012

Cisco Systems Export of Application Information in IPFIX
draft-claise-export-application-info-in-ipfix-09

Status of this Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

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). Not all documents approved by the IESG are a candidate for any level of Internet Standard; see 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/rfc<rfc-no>>.

EDITOR NOTES: The above text is the consensus from the IESG meeting regarding this document. Note that the <RFC-no> must be updated. The text below has been kept for completeness.

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>

<Claise, Aitken, Ben-Dvora> Expires Jan 8 2013

[Page 1]

Internet-Draft <Export of App. Info. in IPFIX > July 2012
This Internet-Draft will expire on January 8, 2013.

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Abstract

This document specifies an Cisco Systems extension to the IPFIX information model specified in [RFC5102] to export application information.

Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

1. Overview.....	5
1.1. IPFIX Documents Overview.....	5
2. Introduction.....	6
2.1. Application Information Use Cases.....	8
3. Terminology.....	8
3.1. New Terminology.....	9
4. applicationId Information Element Specification.....	9
4.1. Existing Classification Engine IDs.....	10
4.2. Selector ID Length per Classification IDs.....	14
4.3. Application Name Options Template Record.....	15
4.4. Resolving IANA L4 Port Discrepancies.....	16
5. Grouping the Applications with the Attributes.....	16
5.1. Options Template Record for the Attribute Values...	18
6. Application Id Examples.....	18
6.1. Example 1: Layer 2 Protocol.....	19
6.2. Example 2: Standardized IANA Layer 3 Protocol.....	20
6.3. Example 3: Proprietary Layer 3 Protocol.....	21
6.4. Example 4: Standardized IANA Layer 4 Port.....	22
6.5. Example 5: Layer 7 Application.....	23
6.6. Example 6: Layer 7 Application with Private Enterprise Number (PEN).....	24
6.7. Example: port Obfuscation.....	26
6.8. Example: Application Name Mapping Options Template.	27
6.9. Example: Attributes Values Options Template Record.	28
7. IANA Considerations.....	29
7.1. New Information Elements.....	29
7.1.1. applicationDescription.....	30
7.1.2. applicationId.....	30
7.1.3. applicationName.....	30
7.1.4. classificationEngineId.....	30
7.1.5. applicationCategoryName.....	33
7.1.6. applicationSubCategoryName.....	33
7.1.7. applicationGroupName.....	34
7.1.8. p2pTechnology.....	34
7.1.9. tunnelTechnology.....	34
7.1.10. encryptedTechnology.....	34
7.2. Classification Engine Ids Registry.....	35
8. Security Considerations.....	35
9. References.....	36
9.1. Normative References.....	36
9.2. Informative References.....	36
10. Acknowledgement.....	38
11. Authors' Addresses.....	39

Internet-Draft	<Export of App. Info. in IPFIX >	July 2012
Appendix A.	Additions to XML Specification of IPFIX	
Information Elements (non normative).....		39
Appendix B.	Port Collisions Tables (non normative).....	45
Appendix C.	Application Registry Example (non normative).....	49

List of Figures and Tables

Figure 1: applicationId Information Element	9
Table 1: Existing Classification Engine IDs	13
Table 2: Selector ID default length per Classification Engine ID	14
Table 3: Application Id Static Attributes	18
Table 4: Different Protocols on UDP and TCP	47
Table 5: Different Protocols on SCTP and TCP	49

1. Overview

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 the IPFIX Architecture [RFC5470], per the requirements defined in RFC 3917 [RFC3917].

The IPFIX Architecture [RFC5470] specifies how IPFIX Data Records and Templates are carried via a congestion-aware transport protocol from IPFIX Exporting Processes to IPFIX Collecting Processes.

IPFIX has a formal description of IPFIX Information Elements, their name, type and additional semantic information, as specified in the IPFIX information model [RFC5102].

In order to gain a level of confidence in the IPFIX implementation, probe the conformity and robustness, and allow interoperability, the Guidelines for IPFIX Testing

Internet-Draft <Export of App. Info. in IPFIX > July 2012
[RFC5471] presents a list of tests for implementers of
compliant Exporting Processes and Collecting Processes.

The Bidirectional Flow Export [RFC5103] specifies a method
for exporting bidirectional flow (biflow) information using
the IP Flow Information Export (IPFIX) protocol, representing
each Biflow using a single Flow Record.

The "Reducing Redundancy in IP Flow Information Export
(IPFIX) and Packet Sampling (PSAMP) Reports" [RFC5473]
specifies a bandwidth saving method for exporting Flow or
packet information, by separating information common to
several Flow Records from information specific to an
individual Flow Record: common Flow information is exported
only once.

2. Introduction

Today service providers and network administrators are
looking for visibility into the packet content rather than
just the packet header. Some network devices Metering
Processes inspect the packet content and identify the
applications that are utilizing the network traffic.
Applications in this context are defined as networking
protocols used by networking processes that exchange
packets between them (such as web applications, peer to
peer applications, file transfer, e-mail applications,
etc.). Applications can be further characterized by other
criteria, some of which are application specific.
Examples include: web application to a specific domain, per
user specific traffic, a video application with a specific
codec, etc...

The application identification is based on several
different methods or even a combination of methods:

1. L2 (Layer 2) protocols (such as ARP (Address Resolution Protocol), PPP (Point-to-Point Protocol), LLDP (Link Layer Discovery Protocol))
2. IP protocols (such as ICMP (Internet Control Message Protocol), IGMP (Internet Group Management Protocol), GRE (Generic Routing Encapsulation))
3. TCP or UDP ports (such as HTTP, Telnet, FTP)
4. Application layer header (of the application to be identified)
5. Packet data content
6. Packets and traffic behavior

Internet-Draft <Export of App. Info. in IPFIX > July 2012
The exact application identification methods are part of the Metering Process internals that aim to provide an accurate identification and minimize false identification. This task requires a sophisticated Metering Process since the protocols do not behave in a standard manner.

1. Applications use port obfuscation where the application runs on different port than the IANA assigned one. For example an HTTP server might run a TCP port 23 (assigned to telnet in [IANA-PORTS])
2. IANA port registries do not accurately reflect how certain ports are "commonly" used today. Some ports are reserved, but the application either never became prevalent or is not in use today.
3. The application behavior and identification logic become more and more complex

For that reason, such Metering Processes usually detect applications based on multiple mechanisms in parallel. Detection based only on port matching might wrongly identify the application. If the Metering Process is capable of detecting applications more accurately, it is considered to be stronger and more accurate.

Similarly, a reporting mechanism that uses L4 port based applications only, such as L4:<known port>, would have similar issues. The reporting system should be capable of reporting the applications classified using all types of mechanisms. In particular applications that do not have any IANA port definition. While a mechanism to export application information should be defined, the L4 port being used must be exported using the destination port (destinationTransportPort at [IANA-IPFIX]) in the corresponding IPFIX record.

This document specifies the Cisco Systems application information encoding (as described in section 4.) to export the application information with the IPFIX protocol [RFC5101].

Applications could be identified at different OSI layers, from layer 2 to layer 7. For example: Link Layer

Internet-Draft <Export of App. Info. in IPFIX > July 2012
Distribution Protocol (LLDP) [LLDP] can be identified in
layer 2, ICMP can be identified in layer 3 [IANA-PROTO],
HTTP can be identified in layer 4 [IANA-PORTS], and Webex
can be identified in layer 7.

While an ideal solution would be an IANA registry for
applications above (or inside the payload of) the well-
known ports [IANA-PORTS], this solution is not always
possible. Indeed, the specifications for some applications
embedded in the payload are not available. Some reverse
engineering as well as a ubiquitous language for
application identification, would be required conditions to
be able to manage an IANA registry for these types of
applications. Clearly, these are blocking factors.

This document specifies the Cisco Systems application
information encoding. However, the layer 7 application
registry values are out of scope of this document.

2.1. Application Information Use Cases

There are several use cases for application information:

1. Application Visibility

This is one of the main cases for using the application
information. Network administrators are using
application visibility to understand the main network
consumers, network trends and user behavior.

2. Security Functions

Application knowledge is sometimes used in security
functions in order to provide comprehensive functions
such as Application based firewall, URL filtering,
parental control, intrusion detection, etc.

All of the above use cases require exporting application
information to provide the network function itself or to
log the network function operation.

3. Terminology

IPFIX-specific terminology used in this document is defined
in Section 2 of the IPFIX protocol specification [RFC5101].

Internet-Draft <Export of App. Info. in IPFIX > July 2012
 As in [RFC5101], these IPFIX-specific terms have the first
 letter of a word capitalized when used in this document.

3.1. New Terminology

Application Id

A unique identifier for an application.

When an application is detected, the most granular
 application is encoded in the Application Id.

4. applicationId Information Element Specification

This document specifies the applicationId Information
 Element, which is a single field composed of two parts:

1. 8 bits of Classification Engine ID. The
 Classification Engine can be considered as a
 specific registry for application assignments.
2. m bits of Selector ID. The Selector ID length
 varies depending on the Classification Engine ID.

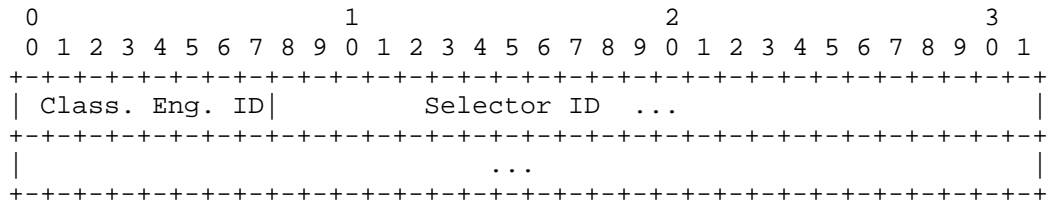


Figure 1: applicationId Information Element

Classification Engine ID

A unique identifier for the engine which determined the
 Selector ID. Thus the Classification Engine ID defines
 the context for the Selector ID.

A unique identifier of the application for a specific Classification Engine ID. Note that the Selector ID length varies depending on the Classification Engine ID.

The Selector ID term is similar in concepts with the selectorId Information Element, specified in the PSAMP Protocol [RFC5476][RFC5477].

4.1. Existing Classification Engine IDs

The following Classification Engine IDs have been allocated:

Name	Value	Description
	0	Invalid.
IANA-L3	1	The Assigned Internet Protocol Number (layer 3 (L3)) is exported in the Selector ID. See [IANA-PROTO].
PANA-L3	2	Proprietary layer 3 definition. An enterprise can export its own layer 3 protocol numbers. The Selector ID has a global significance for all devices from the same enterprise.
IANA-L4	3	The IANA layer 4 (L4) well-known port number is exported in the Selector ID. See [IANA-PORTS]. Note: as an IPFIX flow is unidirectional, it contains the destination port in a flow from the client to the server.
PANA-L4	4	Proprietary layer 4 definition. An enterprise can export its own layer 4 port numbers. The Selector ID has global significance for devices from the

Internet-Draft <Export of App. Info. in IPFIX > July 2012
same enterprise. Example: IPFIX
had the port 4739 pre-assigned in
the IETF draft for years. While
waiting for the RFC and its
associated IANA registration, the
Selector ID 4739 was used with
this PANA-L4.

- | | | |
|------------------|----|--|
| | 5 | Reserved. |
| USER-
Defined | 6 | The Selector ID represents
applications defined by the user
(using CLI, GUI, etc.) based on
the methods described in section
2. The Selector ID has a local
significance per device. |
| | 7 | Reserved. |
| | 8 | Reserved. |
| | 9 | Reserved. |
| | 10 | Reserved. |
| | 11 | Reserved. |
| PANA-L2 | 12 | Proprietary layer 2 (L2)
definition. An enterprise can
export its own layer 2
identifiers. The Selector ID
represents the enterprise's
unique global layer 2
applications. The Selector ID has
a global significance for all
devices from the same enterprise.
Examples include Cisco Subnetwork
Access Protocol (SNAP). |
| PANA-L7 | 13 | Proprietary layer 7 definition.
The Selector ID represents the
enterprise's unique global ID for
the layer 7 applications. The
Selector ID has a global
significance for all devices from
the same enterprise. This
Classification Engine Id is used |

Internet-Draft <Export of App. Info. in IPFIX > July 2012
when the application registry is
owned by the Exporter
manufacturer (referred to as the
"enterprise" in this document).

14 Reserved.

15 Reserved.

16 Reserved.

17 Reserved.

ETHERTYPE 18 The Selector ID represents the
well-known Ethertype. See
[ETHERTYPE]. Note that the
Ethertype is usually expressed in
hexadecimal. However, the
corresponding decimal value is
used in this Selector ID.

LLC 19 The Selector ID represents the
well-known IEEE 802.2 Link Layer
Control (LLC) Destination Service
Access Point (DSAP). See [LLC].
Note that LLC DSAP is usually
expressed in hexadecimal.
However, the corresponding
decimal value is used in this
Selector ID.

PANA-L7-
PEN 20 Proprietary layer 7 definition,
including a Private Enterprise
Number (PEN) [PEN] to identify
that the application registry
being used is not owned by the
Exporter manufacturer (referred
to as the "enterprise" in this
document, and identified by the
PEN), or to identify the original
enterprise in the case of a
mediator or 3rd party device. The
Selector ID represents the
enterprise unique global ID for
the layer 7 applications. The
Selector ID has a global
significance for all devices from

Internet-Draft <Export of App. Info. in IPFIX > July 2012
the same enterprise.

21 to
255 Available (255 is the maximum
Engine ID)

Table 1: Existing Classification Engine IDs

"PANA = Proprietary Assigned Number Authority". In other words, an enterprise specific version of IANA for internal IDs.

The PANA-L7 Classification Engine ID SHOULD be used when the application registry is owned by the Exporter manufacturer, referred to as the "enterprise" in this document, and identified by the PEN. Even if the application registry is owned by the Exporter manufacturer, the PANA-L7-PEN MAY be used, specifying the manufacturer.

The mechanism for the Collector to know about Exporter PEN is out of scope of this document. Possible tracks are: SNMP polling, an Options Template export, hardcoded value, etc.

An Exporter may classify the application according to another vendor's application registry. E.g., an IPFIX Mediator [RFC6183] may need to re-export applications received from different Exporters using different PANA-L7 application registries. For example, X's IPFIX Mediator aggregates traffic from some Exporters which report enterprise Y applications and other Exporters which report enterprise Z applications. Or, X's device implements enterprise Y's application classifications. In these cases, the PANA-L7-PEN Classification Engine MUST be used, which allows the original enterprise ID to be reported. The ID of the enterprise which defined the application ID is identified by the enterprise's PEN. An example is displayed in section 6.6.

Note that the the PANA-L7 Classification Engine ID is also used for resolving IANA L4 port Discrepancies (see Section 4.4)

The list in table 1 is maintained by IANA thanks to the registry within the classificationEngineId Information Element. See the "IANA Considerations" section. The

Internet-Draft <Export of App. Info. in IPFIX > July 2012
Classification Engine Id is part of the Application Id encoding, so the classificationEngineId Information Element is currently not required by the specifications in this document. However, this Information Element was created for completeness, as it was anticipated that this Information Element will be required in the future.

4.2. Selector ID Length per Classification IDs

As the Selector Id part of the Application Id is variable based on the Classification Engine ID value, the applicationId SHOULD be encoded in a variable-length Information Element [RFC5101] for the IPFIX export.

The following table displays the Selector ID default length for the different Classification Engine IDs.

Classification Engine ID Name	Selector ID default length (in bytes)
IANA-L3	1
PANA-L3	1
IANA-L4	2
PANA-L4	2
USER-Defined	3
PANA-L2	5
PANA-L7	3
ETHERTYPE	2
LLC	1
PANA-L7-PEN	3 (*)

Table 2: Selector ID default length per Classification Engine ID

(*) There is an extra 4 bytes for the PEN. However, the PEN is not considered part of the Selector ID.

Internet-Draft <Export of App. Info. in IPFIX > July 2012
 If a legacy protocol such as NetFlow version 9 [RFC3954] is used, and this protocol doesn't support variable length Information Elements, then either multiple Template Records (one per applicationId length), or a single Template Record corresponding to the maximum sized applicationId MUST be used.

Application Ids MAY be encoded in a smaller number of bytes, following the same rules as for the IPFIX Reduced Size Encoding [RFC5101].

Application Ids MAY be encoded with a larger length. For example, a normal IANA L3 protocol encoding would take 2 bytes since the Selector ID represents the protocol field from the IP header encoded in one byte. However, an IANA L3 protocol encoding may be encoded with 3 bytes. In this case, the Selector ID value MUST always be encoded in the least significant bits as shown in Figure 2.

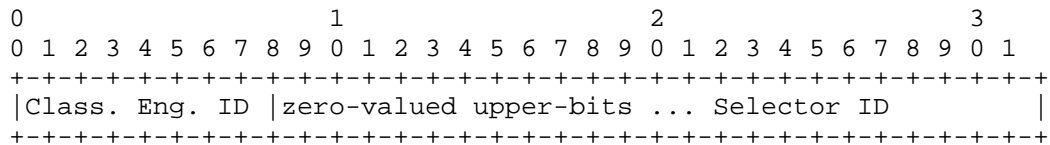


Figure 2: Selector ID encoding

4.3. Application Name Options Template Record

For Classification Engines which specify locally unique Application Ids (which means unique per engine and per router), an Options Template Record (see [RFC5101]) MUST be used to export the correspondence between the Application Id, the Application Name, and the Application Description.

For Classification Engines which specify globally unique Application Ids, an Options Template Record MAY be used to export the correspondence between the Application Id, the Application Name and the Application Description, unless the mapping is hardcoded in the Collector, or known out of band (for example, by polling a MIB).

An example Options Template is shown in section 6.8.

Internet-Draft <Export of App. Info. in IPFIX > July 2012
Enterprises may assign company-wide Application Id values for the PANA-L7 Classification Engine. In this case, a possible optimization for the Collector is to keep the mappings between the Application Ids and the Application Names per enterprise, as opposed to per Exporter.

4.4. Resolving IANA L4 Port Discrepancies

Even though the IANA L4 ports usually point to the same protocols for both UDP, TCP or other transport types, there are some exceptions, as mentioned in the Appendix B.

Instead of imposing the transport protocol (UDP/TCP/SCTP/etc.) in the scope of the "Application Name Options Template Record" (section 6.8.) for all applications (on top of having the transport protocol as key-field in the Flow Record definition), the convention is that the L4 application is always TCP related. So, whenever the Collector has a conflict in looking up IANA, it would choose the TCP choice. As a result, the UDP L4 applications from Table 3 and the SCTP L4 applications from Table 4 are assigned in the PANA_L7 Application Id range, i.e. under Classification Engine ID 13.

Currently, there are no discrepancies between the well known ports for TCP and DCCP.

5. Grouping the Applications with the Attributes

Due to the high number of different Application Ids, Application Ids MAY be categorized into groups. This offers the benefits of easier reporting and action, such as QoS policies. Indeed, most applications with the same characteristics should be treated the same way; for example, all video traffic.

Attributes are statically assigned per Application Id and are independent of the traffic. The attributes are listed below:

Name	Description
Category	An attribute that provides a first level categorization for each Application Id. Examples include:

Internet-Draft	<Export of App. Info. in IPFIX > July 2012 browsing, email, file-sharing, gaming, instant messaging, voice- and-video, etc... The category attribute is encoded by the ApplicationCategoryName Information Element.
Sub-Category	An attribute that provides a second level categorization for each Application Id. Examples include: backup-systems, client-server, database, routing-protocol, etc... The sub-category attribute is encoded by the ApplicationSubCategoryName Information Element.
Application- Group	An attribute that groups multiple Application Ids that belong to the same networking application. For example, the ftp-group contain the ftp-data (port 20), ftp (port 20), ni-ftp (port 47), sftp (port 115), bftp (port 152), ftp-agent(port 574), ftps-data (port 989). The application-group attribute is encoded by the ApplicationGroupName Information Element.
P2P-Technology	Specifies if the Application Id is based on peer-to-peer technology. The P2P-technology attribute is encoded by the p2pTechnology Information Element.
Tunnel- Technology	Specifies if the Application Id is used as a tunnel technology. The tunnel-technology attribute is encoded by the tunnelTechnology Information Element.
Encrypted	Specifies if the Application Id is an encrypted networking protocol. The encrypted attribute is encoded by the encryptedTechnology Information Element.

Every application is assigned to one ApplicationCategoryName, one ApplicationSubCategoryName, one ApplicationGroupName, has one p2pTechnology, one tunnelTechnology, and one encryptedTechnology. These new Information Elements are specified in the IANA Consideration Section 7.1. 7.1.

Maintaining the attribute values in IANA seems impossible to realize. Therefore the attribute values per application are enterprise specific.

5.1. Options Template Record for the Attribute Values

An Options Template Record (see [RFC5101]) SHOULD be used to export the correspondence between each Application Id and its related Attribute values. An alternative way for the Collecting Process to learn the correspondence is to populate these mappings out of band, for example, by loading a CSV file containing the correspondence table.

The Attributes Option Template contains the ApplicationId as a scope field, followed by the ApplicationCategoryName, the ApplicationSubCategoryName, the ApplicationGroupName, the p2pTechnology, the tunnelTechnology, and the encryptedTechnology Information Elements.

A list of attributes may conveniently be exported using a subTemplateList per [RFC6313].

An example is given in section 6.9.

6. Application Id Examples

The following examples are created solely for the purpose of illustrating how the extensions proposed in this document are encoded.

The list of Classification Engine IDs in Table 1 shows that the layer 2 Classification Engine IDs are 12 (PANA-L2), 18, (Ethertype) and 19 (LLC).

From the Ethertype list, LLDP [LLDP] has the Selector ID value 0x88CC, so 35020 in decimal:

NAME	Selector ID
LLDP	35020

So, in the case of LLDP, the Classification Engine ID is 18 (LLC) while the Selector ID has the value 35020.

Per section 4. , the applicationId Information Element, is a single field composed of 8 bits of Classification Engine ID, followed by m bits of Selector ID.

Therefore the Application Id is encoded as:

0								1								2																							
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3																
+-----+-----+-----+-----+-----+-----+-----+-----+																																							
								18																35020															
+-----+-----+-----+-----+-----+-----+-----+-----+																																							

So the Application Id has the decimal value of 1214668. The format '18..35020' is used for simplicity in the examples below, to clearly express that two components of the Application ID.

The Exporting Process creates a Template Record with a few Information Elements: amongst other things, the Application Id. For example:

- applicationId (key field)
- octetTotalCount (non key field)

For example, a Flow Record corresponding to the above Template Record may contain:

```
{ applicationId='18..35020',  
  octetTotalCount=123456 }
```

Internet-Draft <Export of App. Info. in IPFIX > July 2012
 The Collector has all the required information to determine that the application is LLDP, because the Application Id uses a global and well known registry, i.e. the Ethertype. The Collector can determine which application is represented by the Application Id by loading the registry out of band.

6.2. Example 2: Standardized IANA Layer 3 Protocol

From the list of Classification Engine IDs in Table 1, the IANA layer 3 Classification Engine ID (IANA-L3) is 1.

From the list of IANA layer 3 protocols (see [IANA-PROTO]), ICMP has the value 1:

Decimal	Keyword	Protocol	Reference
1	ICMP	Internet Control Message	[RFC792]

So in the case of the standardized IANA layer 3 protocol ICMP, the Classification Engine ID is 1, and the Selector ID has the value of 1.

Therefore the Application Id is encoded as:

```

      0                               1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|           1           |           1           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

So the Application Id has the value of 257. The format '1..1' is used for simplicity in the examples below.

The Exporting Process creates a Template Record with a few Information Elements: amongst other things, the Application Id. For example:

- sourceIPv4Address (key field)
- destinationIPv4Address (key field)
- ipDiffServCodePoint (key field)
- applicationId (key field)
- octetTotalCount (non key field)

Internet-Draft <Export of App. Info. in IPFIX > July 2012
For example, a Flow Record corresponding to the above
Template Record may contain:

```
{ sourceIPv4Address=192.0.2.1,
  destinationIPv4Address=192.0.2.2,
  ipDiffServCodePoint=0,
  applicationId='1..1',
  octetTotalCount=123456 }
```

The Collector has all the required information to determine that the application is ICMP, because the Application Id uses a global and well know registry, ie the IANA L3 protocol number.

6.3. Example 3: Proprietary Layer 3 Protocol

Assume that a enterprise has specified a new layer 3 protocol called "foo".

From the list of Classification Engine IDs in Table 1, the proprietary layer 3 Classification Engine ID (PANA-L3) is 2.

A global registry within the enterprise specifies that the "foo" protocol has the value 90:

Protocol	Protocol Id
foo	90

So, in the case of the layer 3 protocol foo specified by this enterprise, the Classification Engine ID is 2, and the Selector ID has the value of 90.

Therefore the Application Id is encoded as:

```

0                                     1
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|           2           |           90           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

So the Application Id has the value of 602. The format '2..90' is used for simplicity in the examples below.

Internet-Draft <Export of App. Info. in IPFIX > July 2012
 The Exporting Process creates a Template Record with a few
 Information Elements: amongst other things, the Application
 Id. For example:

- sourceIPv4Address (key field)
- destinationIPv4Address (key field)
- ipDiffServCodePoint (key field)
- applicationId (key field)
- octetTotalCount (non key field)

For example, a Flow Record corresponding to the above
 Template Record may contain:

```
{ sourceIPv4Address=192.0.2.1,
  destinationIPv4Address=192.0.2.2,
  ipDiffServCodePoint=0,
  applicationId='2..90',
  octetTotalCount=123456 }
```

Along with this Flow Record, a new Options Template Record
 would be exported, as shown in Section 6.8.

6.4. Example 4: Standardized IANA Layer 4 Port

From the list of Classification Engine IDs in Table 1, the
 IANA layer 4 Classification Engine ID (PANA-L3) is 3.

From the list of IANA layer 4 ports (see [IANA-PORTS]),
 SNMP has the value 161:

Keyword	Decimal	Description
snmp	161/tcp	SNMP
snmp	161/udp	SNMP

So in the case of the standardized IANA layer 4 SNMP port,
 the Classification Engine ID is 3, and the Selector ID has
 the value of 161.

Therefore the Application Id is encoded as:

```

0                                     1
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3
+---+---+---+---+---+---+---+---+---+---+---+---+
|           3           |           161           |
+---+---+---+---+---+---+---+---+---+---+---+---+
```

Internet-Draft <Export of App. Info. in IPFIX > July 2012
So the Application Id has the value of 196769. The format
'3..161' is used for simplicity in the examples below.

The Exporting Process creates a Template Record with a few
Information Elements: amongst other things, the Application
Id. For example:

- sourceIPv4Address (key field)
- destinationIPv4Address (key field)
- protocol (key field)
- ipDiffServCodePoint (key field)
- applicationId (key field)
- octetTotalCount (non key field)

For example, a Flow Record corresponding to the above
Template Record may contain:

```
{ sourceIPv4Address=192.0.2.1,  
  destinationIPv4Address=192.0.2.2,  
  protocol=17, ipDiffServCodePoint=0,  
  applicationId='3..161',  
  octetTotalCount=123456 }
```

The Collector has all the required information to determine
that the application is SNMP, because the Application Id
uses a global and well know registry, ie the IANA L4
protocol number.

6.5. Example 5: Layer 7 Application

In this example, the Metering Process has observed some
Webex traffic.

From the list of Classification Engine IDs in Table 1, the
layer 7 unique Classification Engine ID (PANA-L7) is 13.

Suppose that the Metering Process returns the ID 10000 for
Webex traffic.

So, in the case of this Webex application, the
Classification Engine ID is 13 and the Selector ID has the
value of 10000.

Therefore the Application Id is encoded as:

```

Internet-Draft  <Export of App. Info. in IPFIX >  July 2012
0               1               2               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|           13           |           10000           |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

So the Application Id has the value of 218113808. The format '13..10000' is used for simplicity in the examples below.

The Exporting Process creates a Template Record with a few Information Elements: amongst other things, the Application Id. For example:

- sourceIPv4Address (key field)
- destinationIPv4Address (key field)
- ipDiffServCodePoint (key field)
- applicationId (key field)
- octetTotalCount (non key field)

For example, a Flow Record corresponding to the above Template Record may contain:

```

{ sourceIPv4Address=192.0.2.1,
  destinationIPv4Address=192.0.2.2,
  ipDiffServCodePoint=0,
  applicationId='13..10000',
  octetTotalCount=123456 }

```

The 10000 value is globally unique for the enterprise, so that the Collector can determine which application is represented by the Application Id by loading the registry out of band.

Along with this Flow Record, a new Options Template Record would be exported, as shown in Section 6.8.

6.6. Example 6: Layer 7 Application with Private Enterprise Number (PEN)

In this example, the layer 7 Webex traffic from Example 5 above have been classified by enterprise X. The exported records have been received by enterprise Y's mediation device, which wishes to forward them to a top level Collector.

Internet-Draft <Export of App. Info. in IPFIX > July 2012

In order for the top level Collector to know that the records were classified by enterprise X, the enterprise Y mediation device must report the records using the PANA-L7-PEN Classification Engine ID with enterprise X's Private Enterprise Number.

The PANA-L7-PEN Classification Engine ID is 20, and enterprise X's Selector ID for Webex traffic has the value of 10000.

Therefore the Application Id is encoded as:

```

0                               1                               2                               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|           20           |           enterprise ID = X           |...|
+-----+-----+-----+-----+-----+-----+-----+-----+
|...Ent.ID.contd|           10000           |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

The format '20..X..10000' is used for simplicity in the examples below.

The Exporting Process creates a Template Record with a few Information Elements: amongst other things, the Application Id. For example:

- sourceIPv4Address (key field)
- destinationIPv4Address (key field)
- ipDiffServCodePoint (key field)
- applicationId (key field)
- octetTotalCount (non key field)

For example, a Flow Record corresponding to the above Template Record may contain:

```

{ sourceIPv4Address=192.0.2.1,
  destinationIPv4Address=192.0.2.2,
  ipDiffServCodePoint=0,
  applicationId='20..X..10000',
  octetTotalCount=123456 }

```

The 10000 value is globally unique for enterprise X, so that the Collector can determine which application is represented by the Application Id by loading the registry out of band.

Internet-Draft <Export of App. Info. in IPFIX > July 2012
 Along with this Flow Record, a new Options Template Record
 would be exported, as shown in Section 6.8.

6.7. Example: port Obfuscation

For example, an HTTP server might run on a TCP port 23
 (assigned to telnet in [IANA-PORTS]). If the Metering
 Process is capable of detecting HTTP in the same case, the
 Application Id representation must contain HTTP. However,
 if the reporting application wants to determine whether the
 default HTTP port 80 or 8080 was used, the destination port
 (destinationTransportPort at [IANA-IPFIX]) must also be
 exported in the corresponding IPFIX record.

In the case of a standardized IANA layer 4 port, the
 Classification Engine ID (PANA-L4) is 3, and the Selector
 ID has the value of 80 for HTTP (see [IANA-PORTS]).
 Therefore the Application Id is encoded as:

0	1	2						
0 1 2 3 4 5 6 7 8 9	0 1 2 3 4 5 6 7 8 9	0 1 2 3						
+-----+								
3 80								
+-----+								

The Exporting Process creates a Template Record with a few
 Information Elements: amongst other things, the Application
 Id. For example:

- sourceIPv4Address (key field)
- destinationIPv4Address (key field)
- protocol (key field)
- destinationTransportPort (key field)
- applicationId (key field)
- octetTotalCount (non key field)

For example, a Flow Record corresponding to the above
 Template Record may contain:

```
{ sourceIPv4Address=192.0.2.1,
  destinationIPv4Address=192.0.2.2,
  protocol=17,
  destinationTransportPort=23,
  applicationId='3..80',
  octetTotalCount=123456 }
```

Internet-Draft <Export of App. Info. in IPFIX > July 2012
The Collector has all the required information to determine
that the application is HTTP, but runs on port 23.

6.8. Example: Application Name Mapping Options Template

Along with the Flow Records shown in the above examples, a new Options Template Record should be exported to express the Application Name and Application Description associated with each Application Id.

The Options Template Record contains the following Information Elements:

1. Scope = applicationId.

From RFC 5101: "The scope, which is only available in the Options Template Set, gives the context of the reported Information Elements in the Data Records."

2. applicationName.

3. applicationDescription.

The Options Data Record associated with the examples above would contain, for example:

```
{ scope=applicationId='2..90',  
  applicationName="foo",  
  applicationDescription="The foo protocol",  
  
  scope=applicationId='13..10000',  
  applicationName="webex",  
  applicationDescription="Webex application" }  
  
scope=applicationId='20..X..10000',  
applicationName="webex",  
applicationDescription="Webex application" }
```

When combined with the example Flow Records above, these Options Template Records tell the Collector:

1. A flow of 123456 bytes exists from sourceIPv4Address 192.0.2.1 to destinationIPv4address 192.0.2.2 with an

Internet-Draft <Export of App. Info. in IPFIX > July 2012
applicationId of '12..90', which maps to the "foo"
application.

2. A flow of 123456 bytes exists from sourceIPv4Address
192.0.2.1 to destinationIPv4address 192.0.2.2 with an
Application Id of '13..10000', which maps to the "Webex"
application.

3. A flow of 123456 bytes exists from sourceIPv4Address
192.0.2.1 to destinationIPv4address 192.0.2.2 with an
Application Id of '20..PEN..10000', which maps to the
"Webex" application, according to the application registry
from the enterprise X.

6.9. Example: Attributes Values Options Template Record

Along with the Flow Records shown in the above examples, a
new Options Template Record is exported to express the
values of the different attributes related to the
Application Ids.

The Options Template Record would contain the following
Information Elements:

1. Scope = applicationId.

From RFC 5101: "The scope, which is only available
in the Options Template Set, gives the context of
the reported Information Elements in the Data
Records."

2. applicationCategoryName.

3. applicationSubCategoryName.

4. applicationGroupName

5. p2pTechnology

6. tunnelTechnology

7. encryptedTechnology

Internet-Draft <Export of App. Info. in IPFIX > July 2012
The Options Data Record associated with the examples above
would contain, for example:

```
{ scope=applicationId='2..90',  
  applicationCategoryName="foo-category",  
  applicationSubCategoryName="foo-subcategory",  
  applicationGroupName="foo-group",  
  p2pTechnology=NO  
  tunnelTechnology=YES  
  encryptedTechnology=NO
```

When combined with the example Flow Records above, these
Options Template Records tell the Collector:

A flow of 123456 bytes exists from sourceIPv4Address
192.0.2.1 to destinationIPv4address 192.0.2.2 with a DSCP
value of 0 and an applicationId of '12..90', which maps to
the "foo" application. This application can be
characterized by the relevant attributes values.

7. IANA Considerations

7.1. New Information Elements

This document specifies 10 new IPFIX Information Elements:
the applicationDescription, applicationId, applicationName,
classificationEngineId, applicationCategoryName,
applicationSubCategoryName, applicationGroupName,
p2pTechnology, tunnelTechnology, and encryptedTechnology.

New Information Elements to be added to the IPFIX Information
Element registry at [IANA-IPFIX] are listed below.

EDITOR'S NOTE: RFC5102, which explains the IANA
considerations for assigning new Information Elements
mentions. "The value of these identifiers is in the range of
1-32767. Within this range, Information Element identifier
values in the sub-range of 1-127 are compatible with field
types used by NetFlow version 9 [RFC3954]". This is the
reason why some Information Elements have already an assigned
ElementId in the range 1-127, instead of <TBD>. These
Information Elements should anyway follow the IANA
Considerations from RFC5102, i.e. " New assignments for IPFIX
Information Elements will be administered by IANA through
Expert Review review". The reviewer is Nevil Brownlee.

Internet-Draft <Export of App. Info. in IPFIX > July 2012
EDITOR'S NOTE: the XML specification in Appendix A must be
updated with the elementID values allocated below.

RFC-EDITOR/IANA-EDITOR: some entries are already present in
IPFIX-IANA. However, those must be updated with the current
content.

7.1.1. applicationDescription

Name: applicationDescription
Description:
 Specifies the description of an application.
Abstract Data Type: string
Data Type Semantics:
ElementId: 94
Status: current

7.1.2. applicationId

Name: applicationId
Description:
 Specifies an Application Id.
Abstract Data Type: octetArray
Data Type Semantics: identifier
Reference: See section 4. of [EDITORS NOTE: this document]
for the applicationId Information Element Specification.
ElementId: 95
Status: current

7.1.3. applicationName

Name: applicationName
Description:
 Specifies the name of an application.
Abstract Data Type: string
Data Type Semantics:
ElementId: 96
Status: current

7.1.4. classificationEngineId

Name: classificationEngineId
Description:

Internet-Draft <Export of App. Info. in IPFIX > July 2012

A unique identifier for the engine which determined the Selector ID. Thus the Classification Engine ID defines the context for the Selector ID. The Classification Engine can be considered as a specific registry for application assignments.

Initial values for this field are listed below. Further values may be assigned by IANA in the Classification Engine Ids registry.

0 Invalid.

1 IANA-L3: The Assigned Internet Protocol Number (layer 3 (L3)) is exported in the Selector ID. See <http://www.iana.org/assignments/protocol-numbers>.

2 PANA-L3: Proprietary layer 3 definition. An enterprise can export its own layer 3 protocol numbers. The Selector ID has a global significance for all devices from the same enterprise.

3 IANA-L4: The IANA layer 4 (L4) well-known port number is exported in the Selector ID. See [IANA-PORTS]. Note: as an IPFIX flow is unidirectional, it contains the destination port in a flow from the client to the server.

4 PANA-L4: Proprietary layer 4 definition. An enterprise can export its own layer 4 port numbers. The Selector ID has global significance for devices from the same enterprise. Example: IPFIX had the port 4739 pre-assigned in the IETF draft for years. While waiting for the RFC and its associated IANA registration, the Selector ID 4739 was used with this PANA-L4.

5 Reserved

6 USER-Defined: The Selector ID represents applications defined by the user (using CLI, GUI, etc.) based on the methods described in section 2. The Selector ID has a local significance per device.

7 Reserved

8 Reserved

9 Reserved

10 Reserved

11 Reserved

12 PANA-L2: Proprietary layer 2 (L2) definition. An enterprise can export its own layer 2 identifiers. The Selector ID represents the enterprise's unique global layer 2 applications. The Selector ID has a global significance for all devices from the same enterprise. Examples include Cisco Subnetwork Access Protocol (SNAP).

13 PANA-L7: Proprietary layer 7 definition. The Selector ID represents the enterprise's unique global ID for the layer 7 applications. The Selector ID has a global significance for all devices from the same enterprise. This Classification Engine Id is used when the application registry is owned by the Exporter manufacturer (referred to as the "enterprise" in this document).

14 Reserved

15 Reserved

16 Reserved

17 Reserved

18 ETHERTYPE: The Selector ID represents the well-known Ethertype. See [ETHERTYPE]. Note that the Ethertype is usually expressed in hexadecimal. However, the corresponding decimal value is used in this Selector ID.

19 LLC: The Selector ID represents the well-known IEEE 802.2 Link Layer Control (LLC) Destination Service Access Point (DSAP). See [LLC]. Note that LLC DSAP is usually expressed in hexadecimal. However, the corresponding decimal value is used in this Selector ID.

Internet-Draft <Export of App. Info. in IPFIX > July 2012

20 PANA-L7-PEN: Proprietary layer 7 definition, including a Private Enterprise Number (PEN) [PEN] to identify that the application registry being used is not owned by the Exporter manufacturer (referred to as the "enterprise" in this document, and identified by the PEN), or to identify the original enterprise in the case of a mediator or 3rd party device. The Selector ID represents the enterprise unique global ID for the layer 7 applications. The Selector ID has a global significance for all devices from the same enterprise.

Some values (5, 7, 8, 9, 10, 11, 14, 15, 16, and 17), are reserved to be compliant with existing implementations already using the classificationEngineId.

Abstract Data Type: unsigned8
Data Type Semantics: identifier
ElementId: 101
Status: current

7.1.5. applicationCategoryName

Name: applicationCategoryName
Description:
An attribute that provides a first level categorization for each Application Id.
Abstract Data Type: string
Data Type Semantics:
ElementId: <to be assigned>
Status: current

7.1.6. applicationSubCategoryName

Name: applicationSubCategoryName
Description:
An attribute that provides a second level categorization for each Application Id.
Abstract Data Type: string
Data Type Semantics:
ElementId: <to be assigned>
Status: current

7.1.7. applicationGroupName

Name: applicationGroupName

Description:

An attribute that groups multiple Application Ids that belong to the same networking application.

Abstract Data Type: string

Data Type Semantics:

ElementId: <to be assigned>

Status: current

7.1.8. p2pTechnology

Name: p2pTechnology

Description:

Specifies if the Application Id is based on peer-to-peer technology. Possible values are: { "yes", "y", 1 }, { "no", "n", 2 } and { "unassigned" , "u", 0 }.

Abstract Data Type: string

Data Type Semantics:

ElementId: 288

Status: current

7.1.9. tunnelTechnology

Name: tunnelTechnology

Description:

Specifies if the Application Id is used as a tunnel technology.

Possible values are: { "yes", "y", 1 }, { "no", "n", 2 } and

{ "unassigned" , "u", 0 }.

Abstract Data Type: string

Data Type Semantics:

ElementId: 289

Status: current

7.1.10. encryptedTechnology

Name: encryptedTechnology

Description:

Specifies if the Application Id is an encrypted networking

Internet-Draft <Export of App. Info. in IPFIX > July 2012
protocol. Possible values are: { "yes", "y", 1 },
{ "no", "n", 2 } and { "unassigned", "u", 0 }.
Abstract Data Type: string
Data Type Semantics:
ElementId: 290
Status: current

7.2. Classification Engine Ids Registry

The Information Element #101, named `classificationEngineId`, carries information about the context for the Selector ID, and can be considered as a specific registry for application assignments. For ensuring extensibility of this information, IANA has created a new registry for Classification Engine Ids and filled it with the initial list from the description Information Element #101, `classificationEngineId`.

New assignments for Classification Engine Ids will be administered by IANA through Expert Review [RFC5226], i.e., review by one of a group of experts designated by an IETF Area Director. The group of experts must double check the new definitions with already defined Classification Engine Ids for completeness, accuracy, and redundancy. The specification of Classification Engine Ids MUST be published using a well-established and persistent publication medium.

RFC-EDITOR: this should be assigned similarly to `mplsTopLabelType` subregistry at <http://www.iana.org/assignments/ipfix/ipfix.xml>

8. Security Considerations

The same security considerations as for the IPFIX Protocol [RFC5101] apply.

As mentioned in Section 2.1. , the application knowledge is useful in security based applications. Security applications may impose supplementary requirements on the export of application information, and these need to be examined on a case by case basis.

9. References

9.1. Normative References

- [RFC2119] S. Bradner, Key words for use in RFCs to Indicate Requirement Levels, BCP 14, RFC 2119, March 1997.
- [RFC5101] Claise, B., Ed., "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.
- [RFC5226] Narten, T., and H. Alverstrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, May 2008.
- [ETHERTYPE] <http://standards.ieee.org/develop/regauth/ethertype/eth.txt>
- [IANA-PORTS] <http://www.iana.org/assignments/port-numbers>
- [IANA-PROTO] <http://www.iana.org/assignments/protocol-numbers>
- [LLC] <http://standards.ieee.org/develop/regauth/llc/public.html>.
- [PEN] <http://www.iana.org/assignments/enterprise-numbers>

9.2. Informative References

- [RFC792] J. Postel, Internet Control Message Protocol, RFC 792, September 1981.
- [RFC3917] Quittek, J., Zseby, T., Claise, B., and S. Zander, Requirements for IP Flow Information Export, RFC 3917, October 2004.

Internet-Draft <Export of App. Info. in IPFIX > July 2012
[RFC3954] B. Claise, "Cisco Systems NetFlow Services Export
Version 9", RFC 3954, October 2004.

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

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

[RFC5471] Schmoll, C., Aitken, P., and B. Claise,
"Guidelines for IP Flow Information Export (IPFIX)
Testing", RFC 5471, 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.

[RFC5476] Claise, B., Ed., "Packet Sampling (PSAMP)
Protocol Specifications", RFC 5476, March 2009.

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

[RFC6183] Kobayashi, A., Claise, B., Muenz, G., and K.
Ishibashi, "IP Flow Information Export (IPFIX)
Mediation: Framework", RFC6183, April 2011

[RFC6313] Claise, B., Dhandapani, G. Aitken, P., and S.
Yates, "Export of Structured Data in IP Flow
Information Export (IPFIX)", RFC6313, July 2011

[LLDP] "IEEE Std 802.1AB-2005, Standard for Local and
metropolitan area networks - Station and Media
Access Control Connectivity Discovery", IEEE Std
802.1AB-2005 IEEE Std, 2005.

[IANA-IPFIX]
<http://www.iana.org/assignments/ipfix/ipfix.xml>

Internet-Draft <Export of App. Info. in IPFIX > July 2012
[CISCO-APPLICATION-REGISTRY]
http://www.cisco.com/go/application_registry

10. Acknowledgement

The authors would like to thank their many colleagues across Cisco Systems who made this work possible. Specifically Patrick Wildi for his time and expertise.

Internet-Draft <Export of App. Info. in IPFIX > July 2012
11. Authors' Addresses

Benoit Claise
Cisco Systems, Inc.
De Kleetlaan 6a b1
Diegem 1813
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

Nir Ben-Dvora
Cisco Systems, Inc.
32 HaMelacha St.,
P.O.Box 8735, I.Z.Sapir
South Netanya, 42504
Israel

Phone: +972 9 892 7187
EMail: nirbd@cisco.com

Appendix A. Additions to XML Specification of IPFIX Information Elements (non normative)

This appendix A contains additions to the machine-readable description of the IPFIX information model coded in XML in Appendix A and Appendix B in [RFC5102]. Note that this appendix is of informational nature, while the text in Section 7. (generated from this appendix) is normative.

The following field definitions are appended to the IPFIX information model in Appendix A of [RFC5102].

```
<field name="applicationDescription"
  dataType="string"
  group="application"
  elementId="94" applicability="all"
status="current">
  <description>
    <paragraph>
      Specifies the description of an application.
    </paragraph>
  </description>
</field>

<field name="applicationId"
  dataType="octetArray"
  group="application"
  dataTypeSemantics="identifier"
  elementId="95" applicability="all"
status="current">
  <description>
    <paragraph>
      Specifies an Application Id.
    </paragraph>
  </description>
  <reference>
    <paragraph>
      See section 4. of [EDITORS NOTE: this document]
      for the applicationId Information Element
      Specification.
    </paragraph>
  </reference>
</field>

<field name="applicationName"
  dataType="string"
  group="application"
  elementId="96" applicability="all"
status="current">
  <description>
    <paragraph>
      Specifies the name of an application.
    </paragraph>
  </description>
</field>

<field name="classificationEngineId"
  dataType="unsigned8"
```

Internet-Draft <Export of App. Info. in IPFIX > July 2012

```
group="application"
dataTypeSemantics="identifier"
elementId="101" applicability="all"
```

```
status="current">
```

```
<description>
```

```
<paragraph>
```

```
0 Invalid.
```

```
1 IANA-L3: The Assigned Internet Protocol Number
(layer 3 (L3)) is exported in the Selector ID.
See http://www.iana.org/assignments/protocol-
numbers.
```

```
2 PANA-L3: Proprietary layer 3 definition. An
enterprise can export its own layer 3 protocol
numbers. The Selector ID has a global
significance for all devices from the same
enterprise.
```

```
3 IANA-L4: The IANA layer 4 (L4) well-known port
number is exported in the Selector ID. See
[IANA-PORTS]. Note: as an IPFIX flow is
unidirectional, it contains the destination
port in a flow from the client to the server.
```

```
4 PANA-L4: Proprietary layer 4 definition. An
enterprise can export its own layer 4 port
numbers. The Selector ID has global
significance for devices from the same
enterprise. Example: IPFIX had the port 4739
pre-assigned in the IETF draft for years.
While waiting for the RFC and its associated
IANA registration, the Selector ID 4739 was
used with this PANA-L4.
```

```
5 Reserved
```

```
6 USER-Defined: The Selector ID represents
applications defined by the user (using CLI,
GUI, etc.) based on the methods described in
section 2. The Selector ID has a local
significance per device.
```


- 7 Reserved
- 8 Reserved
- 9 Reserved
- 10 Reserved
- 11 Reserved
- 12 PANA-L2: Proprietary layer 2 (L2) definition.
An enterprise can export its own layer 2
identifiers. The Selector ID represents the
enterprise's unique global layer 2
applications. The Selector ID has a global
significance for all devices from the same
enterprise. Examples include Cisco Subnetwork
Access Protocol (SNAP).
- 13 PANA-L7: Proprietary layer 7 definition. The
Selector ID represents the enterprise's unique
global ID for the layer 7 applications. The
Selector ID has a global significance for all
devices from the same enterprise. This
Classification Engine Id is used when the
application registry is owned by the Exporter
manufacturer (referred to as the "enterprise"
in this document).
- 14 Reserved
- 15 Reserved
- 16 Reserved
- 17 Reserved
- 18 ETHERTYPE: The Selector ID represents the
well-known Ethertype. See [ETHERTYPE]. Note
that the Ethertype is usually expressed in

Internet-Draft <Export of App. Info. in IPFIX > July 2012
hexadecimal. However, the corresponding
decimal value is used in this Selector ID.

19 LLC: The Selector ID represents the well-known
IEEE 802.2 Link Layer Control (LLC)
Destination Service Access Point (DSAP). See
[LLC]. Note that LLC DSAP is usually expressed
in hexadecimal. However, the corresponding
decimal value is used in this Selector ID.

20 PANA-L7-PEN: Proprietary layer 7 definition,
including a Private Enterprise Number (PEN)
[PEN] to identify that the application
registry being used is not owned by the
Exporter manufacturer (referred to as the
"enterprise" in this document, and identified
by the PEN), or to identify the original
enterprise in the case of a mediator or 3rd
party device. The Selector ID represents the
enterprise unique global ID for the layer 7
applications. The Selector ID has a global
significance for all devices from the same
enterprise.

</paragraph>

</description>

</field>

<field name="applicationCategoryName"
 dataType="string"
 group="application"
 elementId="<to be assigned>"
 applicability="all"
 status="current">

<description>

<paragraph>

 An attribute that provides a first level
categorization
 for each Application Id.

</paragraph>

</description>

</field>

<field name="applicationSubCategoryName"

```
      dataType="string"
      group="application"
      elementId="<to be assigned>"
      applicability="all"
      status="current">
    <description>
      <paragraph>
        An attribute that provides a second level
        categorization for each Application Id.
      </paragraph>
    </description>
  </field>

  <field name="applicationGroupName"
    dataType="string"
    group="application"
    elementId="<to be assigned>"
    applicability="all"
    status="current">
    <description>
      <paragraph>
        An attribute that groups multiple Application Ids
        that belong to the same networking application.
      </paragraph>
    </description>
  </field>

  <field name="p2pTechnology"
    dataType="string"
    group="application"
    elementId="288"
    applicability="all"
    status="current">
    <description>
      <paragraph>
        Specifies if the Application Id is based on peer-
        to-peer technology. Possible values are:
        { "yes", "y", 1 }, { "no", "n", 2 } and
        { "unassigned" , "u", 0 }.
      </paragraph>
    </description>
  </field>

  <field name="tunnelTechnology"
    dataType="string"
    group="application"
    elementId="289"
```

```
    applicability="all"
    status="current">
<description>
  <paragraph>
    Specifies if the Application Id is used as a
    tunnel technology. Possible values are:
    { "yes", "y", 1 }, { "no", "n", 2 } and
    { "unassigned" , "u", 0 }.
  </paragraph>
</description>
</field>

<field name="encryptedTechnology"
  dataType="string"
  group="application"
  elementId="290"
  applicability="all"
  status="current">
<description>
  <paragraph>
    Specifies if the Application Id is an encrypted
    networking protocol. Possible values are:
    { "yes", "y", 1 }, { "no", "n", 2 } and
    { "unassigned" , "u", 0 }.
  </paragraph>
</description>
</field>
```

Appendix B. Port Collisions Tables (non normative)

The following table lists the 10 ports that have different protocols assigned for TCP and UDP (at the time of writing this document):

exec	512/tcp	remote process execution; authentication performed using passwords and UNIX login names
comsat/biff	512/udp	used by mail system to notify users of new mail received; currently receives messages only from processes on the same

Internet-Draft <Export of App. Info. in IPFIX > July 2012
machine

login telnet; authentication	513/tcp	remote login a la automatic performed based on privileged port numbers and distributed data bases which identify
who	513/udp	"authentication domains" maintains data bases showing who's logged in to machines on a local net and the load average of the machine
shell	514/tcp	cmd like exec, but automatic authentication is performed as for login server
syslog	514/udp	
oob-ws-https	664/tcp	DMTF out-of-band secure web services management protocol Jim Davis
<jim.davis@wbemsolutions.com> June 2007		
asf-secure-rmcp	664/udp	ASF Secure Remote Management and Control Protocol
rfile	750/tcp	
kerberos-iv	750/udp	kerberos version iv
submit	773/tcp	
notify	773/udp	
rpasswd	774/tcp	
acmaint_dbd	774/udp	
entomb	775/tcp	

busboy	998/tcp	
puparp	998/udp	
garcon	999/tcp	
applix	999/udp	Applix ac

Table 4: Different Protocols on UDP and TCP

The following table lists the 19 ports that have different protocols assigned for TCP and SCTP (at the time of writing this document):

#	3097/tcp	Reserved
itu-bicc-stc	3097/sctp	ITU-T Q.1902.1/Q.2150.3 Greg Sidebottom <gregside&home.com>
#	5090/tcp	<not assigned>
car	5090/sctp	Candidate AR
#	5091/tcp	<not assigned>
cxtcp	5091/sctp	Context Transfer Protocol RFC 4065 - July 2005
#	6704/tcp	Reserved
frc-hp	6704/sctp	ForCES HP (High Priority) channel [RFC5811]
#	6705/tcp	Reserved
frc-mp	6705/sctp	ForCES MP (Medium Priority) channel [RFC5811]
#	6706/tcp	Reserved

Internet-Draft	<Export of App. Info. in IPFIX >	July 2012
frc-lp	6706/sctp	ForCES LP (Low priority) channel [RFC5811]
#	9082/tcp	<not assigned>
lcs-ap	9082/sctp	LCS Application Protocol Kimmo Kymalainen kimmo.kymalainen@etsi.org 04 June 2010
#	9902/tcp	<not assigned>
enrp-sctp-tls	9902/sctp	enrp/tls server channel [RFC5353]
#	11997/tcp	<not assigned>
#	11998/tcp	<not assigned>
#	11999/tcp	<not assigned>
Wmereceiving	11997/sctp	WorldMailExpress
wmedistribution	11998/sctp	WorldMailExpress
wmereporting	11999/sctp	WorldMailExpress Greg Foutz <gregf@adminovation.com> March 2006
#	25471/tcp	<not assigned>
rna	25471/sctp	RNSAP User Adaptation for Iurh Dario S. Tonesi <dario.tonesi@nsn.com> 07 February 2011
#	29118/tcp	Reserved
sgsap	29118/sctp	SGsAP in 3GPP
#	29168/tcp	Reserved
sbcap	29168/sctp	SBcAP in 3GPP
#	29169/tcp	<not assigned>

Internet-Draft	<Export of App. Info. in IPFIX >	July 2012
ihhsctpassoc	29169/sctp	HNBAP and RUA Common Association John Meredith <John.Meredith@etsi.org> 08 September 2009
#	36412/tcp	<not assigned>
s1-control	36412/sctp	S1-Control Plane (3GPP) Kimmo Kymalainen <kimmo.kymalainen@etsi.org> 01 September 2009
#	36422/tcp	<not assigned>
x2-control	36422/sctp	X2-Control Plane (3GPP) Kimmo Kymalainen <kimmo.kymalainen@etsi.org> 01 September 2009
#	36443/tcp	<not assigned>
m2ap	36443/sctp	M2 Application Part Dario S. Tonesi <dario.tonesi@nsn.com> 07 February 2011
#	36444/tcp	<not assigned>
m3ap	36444/sctp	M3 Application Part Dario S. Tonesi <dario.tonesi@nsn.com> 07 February 2011

Table 5: Different Protocols on SCTP and TCP

Appendix C. Application Registry Example (non normative)

A reference to the Cisco Systems assigned numbers for the Application Id and the different attribute assignments can be found at [CISCO-APPLICATION-REGISTRY].

Internet-Draft <Export of App. Info. in IPFIX > July 2012
RFC-EDITOR NOTE: at the time of publication, if [CISCO-
APPLICATION-REGISTRY] is not available, this appendix, and
the [CISCO-APPLICATION-REGISTRY] reference must be removed.

IPFIX Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 7, 2013

B. Trammell
ETH Zurich
A. Wagner
Consecom AG
B. Claise
Cisco Systems, Inc.
July 6, 2012

Flow Aggregation for the IP Flow Information Export (IPFIX) Protocol
draft-ietf-ipfix-a9n-05.txt

Abstract

This document provides a common implementation-independent basis for the interoperable application of the IP Flow Information Export (IPFIX) Protocol to the handling of Aggregated Flows, which are IPFIX Flow representing packets from multiple Original Flows sharing some set of common properties. It does this through a detailed terminology and a descriptive Intermediate Aggregation Process architecture, including a specification of methods for Original Flow counting and counter distribution across intervals.

Status of this Memo

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

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

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

This Internet-Draft will expire on January 7, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	4
1.1. IPFIX Protocol Overview	5
1.2. IPFIX Documents Overview	5
2. Terminology	6
3. Use Cases for IPFIX Aggregation	7
4. Architecture for Flow Aggregation	8
4.1. Aggregation within the IPFIX Architecture	8
4.2. Intermediate Aggregation Process Architecture	12
4.2.1. Correlation and Normalization	14
5. IP Flow Aggregation Operations	15
5.1. Temporal Aggregation through Interval Distribution	15
5.1.1. Distributing Values Across Intervals	16
5.1.2. Time Composition	18
5.1.3. External Interval Distribution	19
5.2. Spatial Aggregation of Flow Keys	19
5.2.1. Counting Original Flows	21
5.2.2. Counting Distinct Key Values	22
5.3. Spatial Aggregation of Non-Key Fields	22
5.3.1. Counter Statistics	22
5.3.2. Derivation of New Values from Flow Keys and non-Key fields	22
5.4. Aggregation Combination	23
6. Additional Considerations and Special Cases in Flow Aggregation	23
6.1. Exact versus Approximate Counting during Aggregation	24
6.2. Delay and Loss introduced by the IAP	24
6.3. Considerations for Aggregation of Sampled Flows	24
6.4. Considerations for Aggregation of Heterogeneous Flows	24
7. Export of Aggregated IP Flows using IPFIX	25
7.1. Time Interval Export	25
7.2. Flow Count Export	26
7.2.1. originalFlowsPresent	26
7.2.2. originalFlowsInitiated	26
7.2.3. originalFlowsCompleted	26
7.2.4. deltaFlowCount	26
7.3. Distinct Host Export	27
7.3.1. distinctCountOfSourceIPAddress	27
7.3.2. distinctCountOfDestinationIPAddress	27

7.3.3.	distinctCountOfSourceIPv4Address	27
7.3.4.	distinctCountOfDestinationIPv4Address	28
7.3.5.	distinctCountOfSourceIPv6Address	28
7.3.6.	distinctCountOfDestinationIPv6Address	28
7.4.	Aggregate Counter Distribution Export	28
7.4.1.	Aggregate Counter Distribution Options Template . . .	29
7.4.2.	valueDistributionMethod Information Element	29
8.	Examples	31
8.1.	Traffic Time-Series per Source	32
8.2.	Core Traffic Matrix	36
8.3.	Distinct Source Count per Destination Endpoint	41
8.4.	Traffic Time-Series per Source with Counter Distribution	43
9.	Security Considerations	45
10.	IANA Considerations	45
11.	Acknowledgments	46
12.	References	46
12.1.	Normative References	46
12.2.	Informative References	46
	Authors' Addresses	47

1. Introduction

The assembly of packet data into Flows serves a variety of different purposes, as noted in the requirements [RFC3917] and applicability statement [RFC5472] for the IP Flow Information Export (IPFIX) protocol [I-D.ietf-ipfix-protocol-rfc5101bis]. Aggregation beyond the flow level, into records representing multiple Flows, is a common analysis and data reduction technique as well, with applicability to large-scale network data analysis, archiving, and inter-organization exchange. This applicability in large-scale situations, in particular, led to the inclusion of aggregation as part of the IPFIX Mediators Problem Statement [RFC5982], and the definition of an Intermediate Aggregation Process in the Mediator framework [RFC6183].

Aggregation is used for analysis and data reduction in a wide variety of applications, for example in traffic matrix calculation, generation of time series data for visualizations or anomaly detection, or data reduction for long-term trending and storage. Depending on the keys used for aggregation, it may additionally have an anonymizing affect on the data: for example, aggregation operations which eliminate IP addresses make it impossible to later identify nodes using those addresses.

Aggregation as defined and described in this document covers the applications defined in [RFC5982], including 5.1 "Adjusting Flow Granularity", 5.4 "Time Composition", and 5.5 "Spatial Composition". However, this document specifies a more flexible architecture for an Intermediate Aggregation Process than that envisioned by the original Mediator work, in Section 4.2. Instead of a focus on these specific limited use cases, the Intermediate Aggregation Process is specified to cover any activity commonly described as "flow aggregation". This architecture is intended to describe any such activity without reference to the specific implementation of aggregation.

An Intermediate Aggregation Process may be applied to data collected from multiple Observation Points, as it is natural to use aggregation for data reduction when concentrating measurement data. This document specifically does not address the protocol issues that arise when combining IPFIX data from multiple Observation Points and exporting from a single Mediator, as these issues are general to IPFIX Mediation; they are therefore treated in detail in the Mediaton Protocol [I-D.ietf-ipfix-mediation-protocol] document.

Since Aggregated Flows as defined in the following section are essentially Flows, the IPFIX protocol [I-D.ietf-ipfix-protocol-rfc5101bis] can be used to export, and the IPFIX File Format [RFC5655] can be used to store, aggregated data "as-is"; there are no changes necessary to the protocol. This

document provides a common basis for the application of IPFIX to the handling of aggregated data, through a detailed terminology, Intermediate Aggregation Process architecture, and methods for Original Flow counting and counter distribution across intervals.

1.1. IPFIX Protocol Overview

In the IPFIX protocol, { type, length, value } tuples are expressed in templates containing { type, length } pairs, specifying which { value } fields are present in data records conforming to the Template, giving great flexibility as to what data is transmitted. Since Templates are sent very infrequently compared with Data Records, this results in significant bandwidth savings. Various different data formats may be transmitted simply by sending new Templates specifying the { type, length } pairs for the new data format. See [I-D.ietf-ipfix-protocol-rfc5101bis] for more information.

The IPFIX Information Element Registry [iana-ipfix-assignments] defines a large number of standard Information Elements which provide the necessary { type } information for Templates. The use of standard elements enables interoperability among different vendors' implementations. Additionally, non-standard enterprise-specific elements may be defined for private use.

1.2. IPFIX Documents Overview

"Specification of the IPFIX Protocol for the Exchange of IP Traffic Flow Information" [I-D.ietf-ipfix-protocol-rfc5101bis] and its associated documents define the IPFIX Protocol, which provides network engineers and administrators with access to IP traffic flow information.

"Architecture for IP Flow Information Export" [RFC5470] defines the architecture for the export of measured IP flow information out of an IPFIX Exporting Process to an IPFIX Collecting Process, and the basic terminology used to describe the elements of this architecture, per the requirements defined in "Requirements for IP Flow Information Export" [RFC3917]. The IPFIX Protocol document [I-D.ietf-ipfix-protocol-rfc5101bis] then covers the details of the method for transporting IPFIX Data Records and Templates via a congestion-aware transport protocol from an IPFIX Exporting Process to an IPFIX Collecting Process.

This document specifies an Intermediate Process which may be applied at an IPFIX Mediator. "IP Flow Information Export (IPFIX) Mediation: Problem Statement" [RFC5982] introduces the concept of IPFIX Mediators, and defines the use cases for which they were designed;

"IP Flow Information Export (IPFIX) Mediation: Framework" [RFC6183] then provides an architectural framework for Mediators. Protocol-level issues (e.g., template and observation domain handling across Mediators) are covered by "Specification of the Protocol for IPFIX Mediation" [I-D.ietf-ipfix-mediation-protocol].

2. Terminology

Terms used in this document that are defined in the Terminology section of the IPFIX Protocol [I-D.ietf-ipfix-protocol-rfc5101bis] document are to be interpreted as defined there.

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].

In addition, this document defines the following terms

Aggregated Flow: A Flow, as defined by [I-D.ietf-ipfix-protocol-rfc5101bis], derived from a set of zero or more original Flows within a defined Aggregation Interval. The primary difference between a Flow and an Aggregated Flow in the general case is that the time interval (i.e., the two-tuple of start and end times) of a Flow is derived from information about the timing of the packets comprising the Flow, while the time interval of an Aggregated Flow is often externally imposed. Note that an Aggregated Flow is defined in the context of an Intermediate Aggregation Process only. Once an Aggregated Flow is exported, it is essentially a Flow as in [I-D.ietf-ipfix-protocol-rfc5101bis] and can be treated as such.

Intermediate Aggregation Process: an Intermediate Process as in [RFC6183] that aggregates records, based upon a set of Flow Keys or functions applied to fields from the record.

Aggregation Interval: A time interval imposed upon an Aggregated Flow. Intermediate Aggregation Processes may use a regular Aggregation Interval (e.g. "every five minutes", "every calendar month"), though regularity is not necessary. Aggregation intervals may also be derived from the time intervals of the Original Flows being aggregated.

Partially Aggregated Flow: A Flow during processing within an Intermediate Aggregation Process; refers to an intermediate data structure during aggregation within the Intermediate Aggregation Process architecture detailed in Section 4.2.

Original Flow: A Flow given as input to an Intermediate Aggregation Process in order to generate Aggregated Flows.

Contributing Flow: An Original Flow that is partially or completely represented within an Aggregated Flow. Each Aggregated Flow is made up of zero or more Contributing Flows, and an Original Flow may contribute to zero or more Aggregated Flows.

Original Exporter: When the Intermediate Aggregation Process is hosted in an IPFIX Mediator, the Original Exporter is the Exporter from which the Original Flows are received.

The terminology presented herein improves the precision of, but does not supersede or contradict the terms related to mediation and aggregation defined in the problem statement [RFC5982] and the framework [RFC6183] documents. Within this document, the terminology defined in this section is to be considered normative.

3. Use Cases for IPFIX Aggregation

Aggregation, as a common data reduction method used in traffic data analysis, has many applications. When used with a regular Aggregation Interval and Original Flows containing timing information, it generates time series data from a collection of Flows with discrete intervals, as in the example in Section 8.1. This time series data is itself useful for a wide variety of analysis tasks, such as generating input for network anomaly detection systems, or driving visualizations of volume per time for traffic with specific characteristics. As a second example, traffic matrix calculation from flow data, as shown in Section 8.2 is inherently an aggregation action, by aggregating the Flow Key down to input or output interface, address prefix, or autonomous system.

Irregular or data-dependent Aggregation Intervals and key aggregation operations can also be used to provide adaptive aggregation of network flow data. Here, full Flow Records can be kept for Flows of interest, while Flows deemed "less interesting" to a given application can be aggregated. For example, in an IPFIX Mediator equipped with traffic classification capabilities for security purposes, potentially malicious Flows could be exported directly, while known-good or probably-good Flows (e.g. normal web browsing) could be exported simply as time series volumes per web server.

Aggregation can also be applied to final analysis of stored Flow data, as shown in the example in Section 8.3. All such aggregation applications in which timing information is not available or not important can be treated as if an infinite Aggregation Interval

applies.

Note that an Intermediate Aggregation Process which removes potentially sensitive information as identified in [RFC6235] may tend to have an anonymising effect on the Aggregated Flows, as well; however, any application of aggregation as part of a data protection scheme should ensure that all the issues raised in [RFC6235] are addressed, specifically Section 4 "Anonymization of IP Flow Data", Section 7.2 "IPFIX-Specific Anonymization Guidelines", and Section 9 "Security Considerations".

While much of the discussion in this document, and all of the examples, apply to the common case that the Original Flows to be aggregated are all of the same underlying type (i.e., are represented with identical or compatible Templates), and that each packet observed by the Metering Process on the far side of the Original Exporter is represented, this is not a necessary assumption. Aggregation can also be applied as part of a technique applying both aggregation and correlation to pull together multiple views of the same traffic from different Observation Points using different Templates. For example, consider a set of applications running at different Observation Points for different purposes -- one generating flows with round-trip-times for passive performance measurement, and one generating billing records. Once correlated, these flows could be used to produce Aggregated Flows containing both volume and performance information together. The correlation and normalization operation described in Section 4.2.1 handles this specific case of correlation. Flow correlation in the general case is outside the scope of this document.

4. Architecture for Flow Aggregation

This section specifies the architecture of the Intermediate Aggregation Process, and how it fits into the IPFIX Architecture.

4.1. Aggregation within the IPFIX Architecture

An Intermediate Aggregation Process could be deployed at any of three places within the IPFIX Architecture. While aggregation is most commonly done within a Mediator which collects Original Flows from an Original Exporter and exports Aggregated Flows, aggregation can also occur before initial export, or after final collection, as shown in Figure 1. The presence of an IAP at any of these points is of course optional.

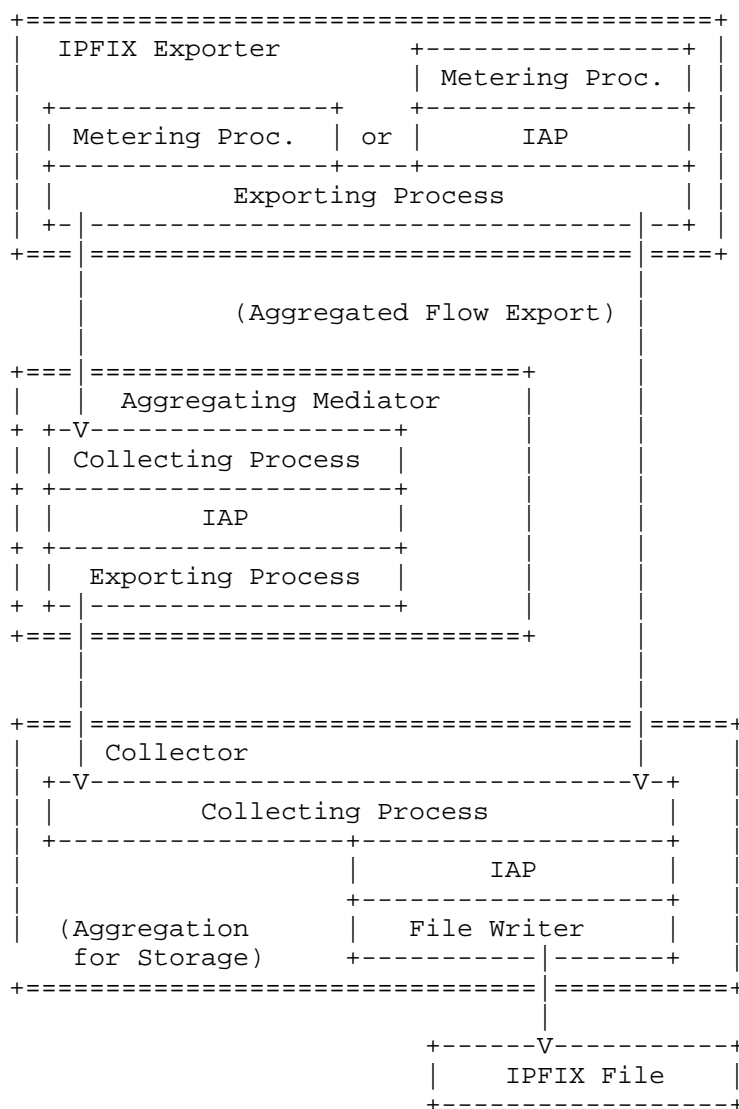


Figure 1: Potential Aggregation Locations

The Mediator use case is further shown in Figures A and B in [RFC6183].

Aggregation can be applied for either intermediate or final analytic purposes. In certain circumstances, it may make sense to export Aggregated Flows directly from an original Exporting Process, for example, if the Exporting Process is applied to drive a time-series

visualization, or when flow data export bandwidth is restricted and flow or packet sampling is not an option. Note that this case, where the Aggregation Process is essentially integrated into the Metering Process, is essentially covered by the IPFIX architecture [RFC5470]: the Flow Keys used are simply a subset of those that would normally be used, and time intervals may be chosen other than those available from the cache policies customarily offered by the Metering Process. A Metering Process in this arrangement MAY choose to simulate the generation of larger Flows in order to generate Original Flow counts, if the application calls for compatibility with an Aggregation Process deployed in a separate location.

In the specific case that an Aggregation Process is employed for data reduction for storage purposes, it can take Original Flows from a Collecting Process or File Reader and pass Aggregated Flows to a File Writer for storage.

Deployment of an Intermediate Aggregation Process within a Mediator [RFC5982] is a much more flexible arrangement. Here, the Mediator consumes Original Flows and produces Aggregated Flows; this arrangement is suited to any of the use cases detailed in Section 3. In a Mediator, Original Flows from multiple sources can also be aggregated into a single stream of Aggregated Flows; the architectural specifics of this arrangement are not addressed in this document, which is concerned only with the aggregation operation itself; see [I-D.ietf-ipfix-mediation-protocol] for details.

The data paths into and out of an Intermediate Aggregation Process are shown in Figure 2.

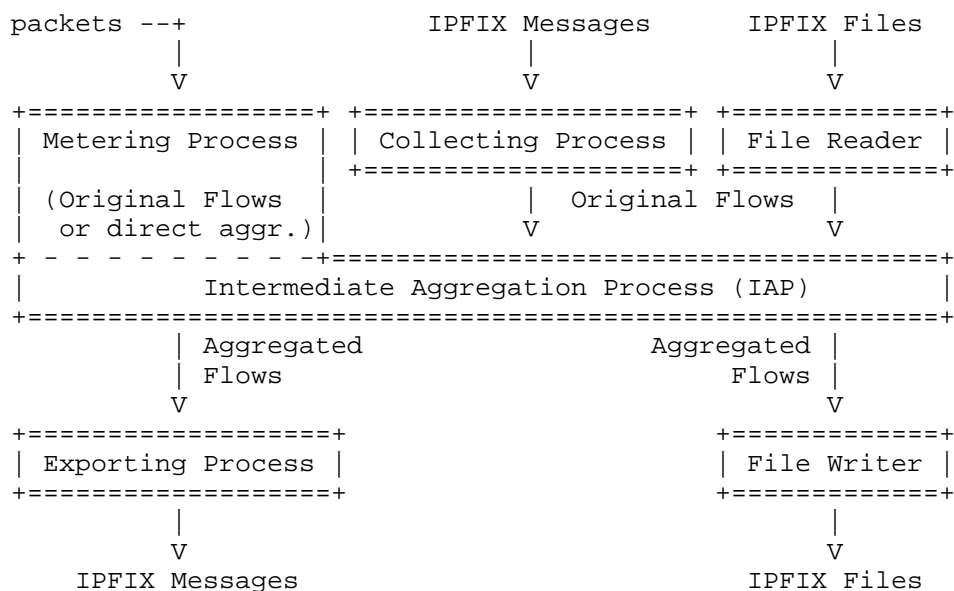


Figure 2: Data paths through the aggregation process

Aggregation may also need to correlate original flows from multiple Metering Processes, each according to a different Template with different Flow Keys and values. This arrangement is shown in Figure 3; in this case, the correlation and normalization operation described in Section 4.2.1 handles merging the Original Flows before

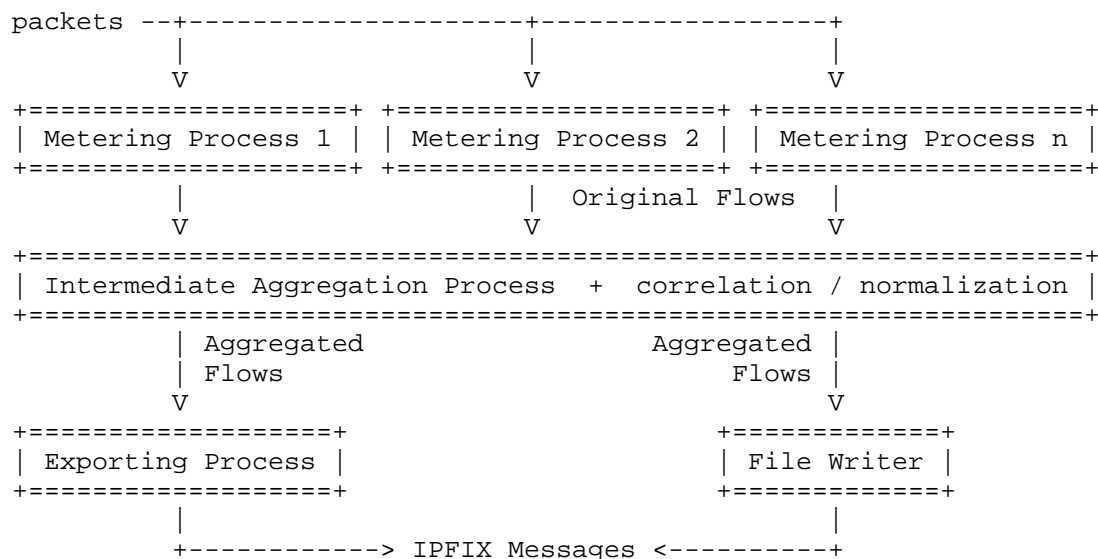


Figure 3: Aggregating Original Flows from multiple Metering Processes

4.2. Intermediate Aggregation Process Architecture

Within this document, an Intermediate Aggregation Process can be seen as hosting a function composed of four types of operations on Partially Aggregated Flows, as illustrated in Figure 4. "Partially Aggregated Flows" as defined in Section 2 are essentially the intermediate results of aggregation, internal to the Intermediate Aggregation Process.

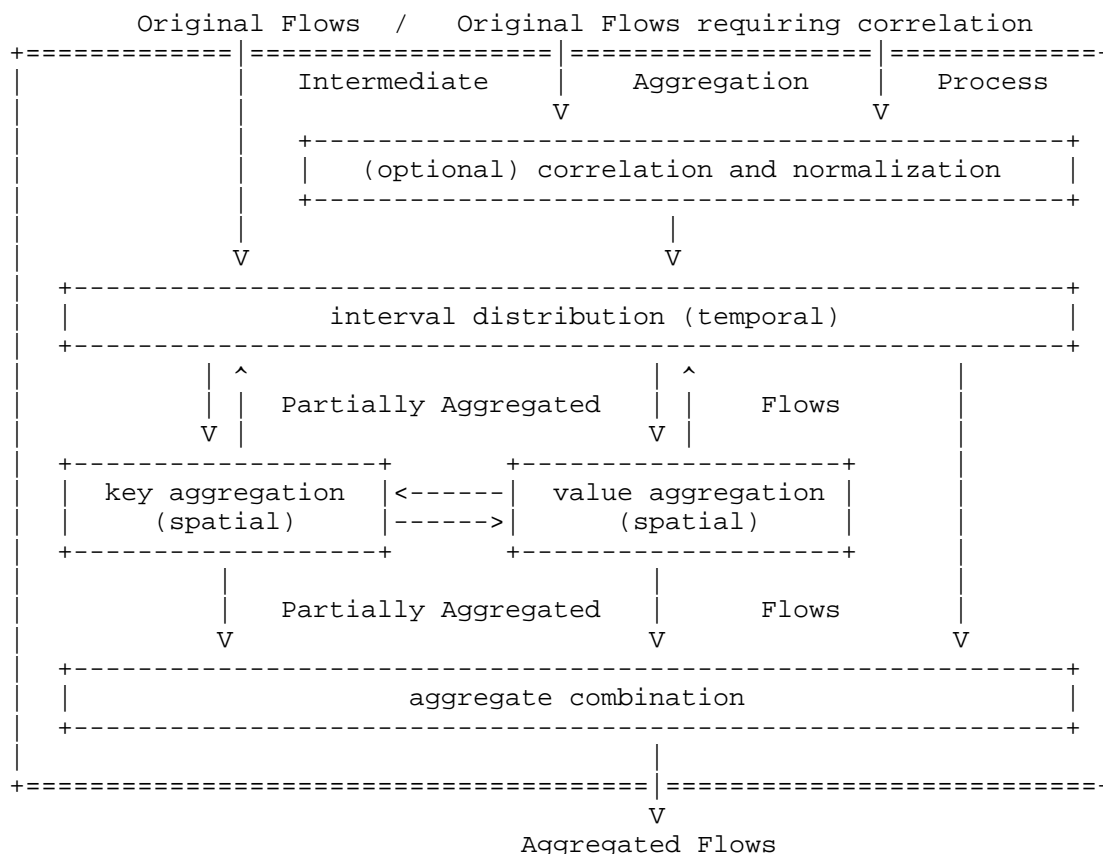


Figure 4: Conceptual model of aggregation operations within an IAP

Interval distribution: a temporal aggregation operation which imposes an Aggregation Interval on the partially Aggregated Flow. This Aggregation Interval may be regular, irregular, or derived from the timing of the Original Flows themselves. Interval distribution is discussed in detail in Section 5.1.

Key aggregation: a spatial aggregation operation which results in the addition, modification, or deletion of Flow Key fields in the Partially Aggregated Flows. New Flow Keys may be derived from existing Flow Keys (e.g., looking up an AS number for an IP address), or "promoted" from specific non-Key fields (e.g., when aggregating Flows by packet count per Flow). Key aggregation can also add new non-Key fields derived from Flow Keys that are deleted during key aggregation; mainly counters of unique reduced keys. Key aggregation is discussed in detail in Section 5.2.

Value aggregation: a spatial aggregation operation which results in the addition, modification, or deletion of non-Key fields in the Partially Aggregated Flows. These non-Key fields may be "demoted" from existing Key fields, or derived from existing Key or non-Key fields. Value aggregation is discussed in detail in Section 5.3.

Aggregate combination: an operation combining multiple partially Aggregated Flows having undergone interval distribution, key aggregation, and value aggregation which share Flow Keys and Aggregation Intervals into a single Aggregated Flow per set of Flow Key values and Aggregation Interval. Aggregate combination is discussed in detail in Section 5.4.

Correlation and normalization: an optional operation, applies when accepting Original Flows from Metering Processes which export different views of essentially the same Flows before aggregation; the details of correlation and normalization are specified in Section 4.2.1, below.

The first three of these operations may be carried out any number of times in any order, either on Original Flows or on the results of one of the operations above, with one caveat: since Flows carry their own interval data, any spatial aggregation operation implies a temporal aggregation operation, so at least one interval distribution step, even if implicit, is required by this architecture. This is shown as the first step for the sake of simplicity in the diagram above. Once all aggregation operations are complete, aggregate combination ensures that for a given Aggregation Interval, set of Flow Key values, and Observation Domain, only one Flow is produced by the Intermediate Aggregation Process.

This model describes the operations within a single Intermediate Aggregation Process, and it is anticipated that most aggregation will be applied within a single process. However, as the steps in the model may be applied in any order and aggregate combination is idempotent, any number of Intermediate Aggregation Processes operating in series can be modeled as a single process. This allows aggregation operations to be flexibly distributed across any number of processes, should application or deployment considerations so dictate.

4.2.1. Correlation and Normalization

When accepting Original Flows from multiple Metering Processes, each of which provides a different view of the Original Flow as seen from the point of view of the IAP, an optional correlation and normalization operation combines each of these single Flow Records into a set of unified partially aggregated Flows before applying

interval distribution. These unified Flows appear as if they had been measured at a single Metering Process which used the union of the set of Flow Keys and non-key fields of all Metering Processes sending Original Flows to the IAP.

Since, due to export errors or other slight irregularities in flow metering, the multiple views may not be completely consistent; normalization involves applying a set of aggregation application specific corrections in order to ensure consistency in the unified Flows.

In general, correlation and normalization should take multiple views of essentially the same Flow, as determined by the configuration of the operation itself, and render them into a single unified Flow. Flows which are essentially different should not be unified by the correlation and normalization operation. This operation therefore requires enough information about the configuration and deployment of Metering Processes from which it correlates Original Flows in order to make this distinction correctly and consistently.

The exact steps performed to correlate and normalize flows in this step are application-, implementation-, and deployment-specific, and will not be further specified in this document.

5. IP Flow Aggregation Operations

As stated in Section 2, an Aggregated Flow is simply an IPFIX Flow generated from Original Flows by an Intermediate Aggregation Process. Here, we detail the operations by which this is achieved within an Intermediate Aggregation Process.

5.1. Temporal Aggregation through Interval Distribution

Interval distribution imposes a time interval on the resulting Aggregated Flows. The selection of an interval is specific to the given aggregation application. Intervals may be derived from the Original Flows themselves (e.g., an interval may be selected to cover the entire time containing the set of all Flows sharing a given Key, as in Time Composition described in Section 5.1.2) or externally imposed; in the latter case the externally imposed interval may be regular (e.g., every five minutes) or irregular (e.g., to allow for different time resolutions at different times of day, under different network conditions, or indeed for different sets of Original Flows).

The length of the imposed interval itself has tradeoffs. Shorter intervals allow higher-resolution aggregated data and, in streaming applications, faster reaction time. Longer intervals generally lead

to greater data reduction and simplified counter distribution. Specifically, counter distribution is greatly simplified by the choice of an interval longer than the duration of longest Original Flow, itself generally determined by the Original Flow's Metering Process active timeout; in this case an Original Flow can contribute to at most two Aggregated Flows, and the more complex value distribution methods become inapplicable.

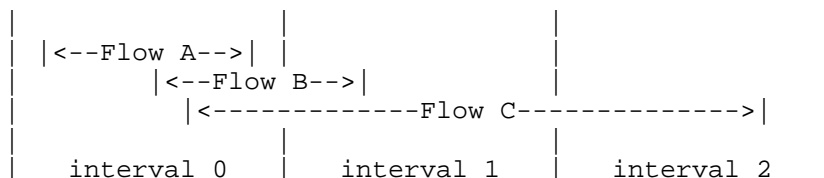


Figure 5: Illustration of interval distribution

In Figure 5, we illustrate three common possibilities for interval distribution as applies with regular intervals to a set of three Original Flows. For Flow A, the start and end times lie within the boundaries of a single interval 0; therefore, Flow A contributes to only one Aggregated Flow. Flow B, by contrast, has the same duration but crosses the boundary between intervals 0 and 1; therefore, it will contribute to two Aggregated Flows, and its counters must be distributed among these Flows, though in the two-interval case this can be simplified somewhat simply by picking one of the two intervals, or proportionally distributing between them. Only Flows like Flow A and Flow B will be produced when the interval is chosen to be longer than the duration of longest Original Flow, as above. More complicated is the case of Flow C, which contributes to more than two Aggregated Flows, and must have its counters distributed according to some policy as in Section 5.1.1.

5.1.1. Distributing Values Across Intervals

In general, counters in Aggregated Flows are treated the same as in any Flow. Each counter is independently calculated as if it were derived from the set of packets in the Original Flow. For the most part, when aggregating Original Flows into Aggregated Flows, this is simply done by summation.

When the Aggregation Interval is guaranteed to be longer than the longest Original Flow, a Flow can cross at most one Interval boundary, and will therefore contribute to at most two Aggregated Flows. Most common in this case is to arbitrarily but consistently choose to account the Original Flow's counters either to the first or the last Aggregated Flow to which it could contribute.

However, this becomes more complicated when the Aggregation Interval is shorter than the longest Original Flow in the source data. In such cases, each Original Flow can incompletely cover one or more time intervals, and apply to one or more Aggregated Flows. In this case, the Aggregation Process must distribute the counters in the Original Flows across the multiple Aggregated Flows. There are several methods for doing this, listed here in roughly increasing order of complexity and accuracy; most of these are necessary only in specialized cases.

End Interval: The counters for an Original Flow are added to the counters of the appropriate Aggregated Flow containing the end time of the Original Flow.

Start Interval: The counters for an Original Flow are added to the counters of the appropriate Aggregated Flow containing the start time of the Original Flow.

Mid Interval: The counters for an Original Flow are added to the counters of a single appropriate Aggregated Flow containing some timestamp between start and end time of the Original Flow.

Simple Uniform Distribution: Each counter for an Original Flow is divided by the number of time intervals the Original Flow covers (i.e., of appropriate Aggregated Flows sharing the same Flow Keys), and this number is added to each corresponding counter in each Aggregated Flow.

Proportional Uniform Distribution: This is like simple uniform distribution, but accounts for the fractional portions of a time interval covered by an Original Flow in the first and last time interval. Each counter for an Original Flow is divided by the number of time units the Original Flow covers, to derive a mean count rate. This rate is then multiplied by the number of time units in the intersection of the duration of the Original Flow and the time interval of each Aggregated Flow.

Simulated Process: Each counter of the Original Flow is distributed among the intervals of the Aggregated Flows according to some function the Aggregation Process uses based upon properties of Flows presumed to be like the Original Flow. For example, Flow Records representing bulk transfer might follow a more or less proportional uniform distribution, while interactive processes are far more bursty.

Direct: The Aggregation Process has access to the original packet timings from the packets making up the Original Flow, and uses these to distribute or recalculate the counters.

A method for exporting the distribution of counters across multiple Aggregated Flows is detailed in Section 7.4. In any case, counters MUST be distributed across the multiple Aggregated Flows in such a way that the total count is preserved, within the limits of accuracy of the implementation. This property allows data to be aggregated and re-aggregated with negligible loss of original count information. To avoid confusion in interpretation of the aggregated data, all the counters for a set of given Original Flows SHOULD be distributed via the same method.

More complex counter distribution methods generally require that the interval distribution process track multiple "current" time intervals at once. This may introduce some delay into the aggregation operation, as an interval should only expire and be available for export when no additional Original Flows applying to the interval are expected to arrive at the Intermediate Aggregation Process.

Note, however, that since there is no guarantee that Flows from the Original Exporter will arrive in any given order, whether for transport-specific reasons (i.e. UDP reordering) or Metering Process implementation-specific reasons, even simpler distribution methods may need to deal with flows arriving in other than start time or end time order. Therefore, the use of larger intervals does not obviate the need to buffer Partially Aggregated Flows within "current" time intervals, to ensure it can accept flow time intervals in any arrival order. More generally, the interval distribution process SHOULD accept flow start and end times in the Original Flows in any reasonable order. The expiration of intervals in interval distribution operations is dependent on implementation and deployment requirements, and SHOULD be made configurable in contexts in which "reasonable order" is not obvious at implementation time. This operation may lead to delay and loss introduced by the IAP, as detailed in Section 6.2.

5.1.2. Time Composition

Time Composition as in Section 5.4 of [RFC5982] (or interval combination) is a special case of aggregation, where interval distribution imposes longer intervals on Flows with matching keys and "chained" start and end times, without any key reduction, in order to join long-lived Flows which may have been split (e.g., due to an active timeout shorter than the actual duration of the Flow.) Here, no Key aggregation is applied, and the Aggregation Interval is chosen on a per-Flow basis to cover the interval spanned by the set of

aggregated Flows. This may be applied alone in order to normalize split Flows, or in combination with other aggregation functions in order to obtain more accurate Original Flow counts.

5.1.3. External Interval Distribution

Note that much of the difficulty of interval distribution at an IAP can be avoided simply by configuring the original Exporters to synchronize the time intervals in the Original Flows with the desired aggregation interval. The resulting Original Flows would then be split to align perfectly with the time intervals imposed during Interval Imposition (i.e., like Flow A in Figure 5), though this may reduce their usefulness for non-Aggregation purposes. This approach allows the Intermediate Aggregation Process to use Start Interval or End Interval distribution, while having equivalent information to that available to Direct interval distribution.

5.2. Spatial Aggregation of Flow Keys

Key aggregation generates a new set of Flow Key values for the Aggregated Flows from the Original Flow Key and non-Key fields in the Original Flows, or from correlation of the Original Flow information with some external source. There are two basic operations here. First, Aggregated Flow Keys may be derived directly from Original Flow Keys through reduction, or the dropping of fields or precision in the Original Flow Keys. Second, Aggregated Flow Keys may be derived through replacement, e.g. by removing one or more fields from the Original Flow and replacing them with fields derived from the removed fields. Replacement may refer to external information (e.g., IP to AS number mappings). Replacement may apply to Flow Keys as well as non-key fields. For example, consider an application which aggregates Original Flows by packet count (i.e., generating an Aggregated Flow for all one-packet Flows, one for all two-packet Flows, and so on). This application would promote the packet count to a Flow Key.

Key aggregation may also result in the addition of new non-Key fields to the Aggregated Flows, namely Original Flow counters and unique reduced key counters; these are treated in more detail in Section 5.2.1 and Section 5.2.2, respectively.

In any key aggregation operation, reduction and/or replacement may be applied any number of times in any order. Which of these operations are supported by a given implementation is implementation- and application-dependent.

Original Flow Keys

src ip4	dst ip4	src port	dst port	proto	tos
retain	mask /24	X	X	X	X
V	V				
src ip4	dst ip4 /24				

Aggregated Flow Keys (by source address and destination class-C)

Figure 6: Illustration of key aggregation by reduction

Figure 6 illustrates an example reduction operation, aggregation by source address and destination class C network. Here, the port, protocol, and type-of-service information is removed from the Flow Key, the source address is retained, and the destination address is masked by dropping the lower 8 bits.

Original Flow Keys

src ip4	dst ip4	src port	dst port	proto	tos
		X	X	X	X
ASN lookup table					
V	V				
src asn	dst asn				

Aggregated Flow Keys (by source and dest ASN)

Figure 7: Illustration of key aggregation by reduction and replacement

Figure 7 illustrates an example reduction and replacement operation, aggregation by source and destination Border Gateway Protocol (BGP) Autonomous System Number (ASN) without ASN information available in the Original Flow. Here, the port, protocol, and type-of-service information is removed from the Flow Keys, while the source and destination addresses are run through an IP address to ASN lookup table, and the Aggregated Flow Keys are made up of the resulting source and destination ASNs.

5.2.1. Counting Original Flows

When aggregating multiple Original Flows into an Aggregated Flow, it is often useful to know how many Original Flows are present in the Aggregated Flow. This document introduces four new information elements in Section 7.2 to export these counters.

There are two possible ways to count Original Flows, which we call here conservative and non-conservative. Conservative flow counting has the property that each Original Flow contributes exactly one to the total flow count within a set of Aggregated Flows. In other words, conservative flow counters are distributed just as any other counter during interval distribution, except each Original Flow is assumed to have a flow count of one. When a count for an Original Flow must be distributed across a set of Aggregated Flows, and a distribution method is used which does not account for that Original Flow completely within a single Aggregated Flow, conservative flow counting requires a fractional representation.

By contrast, non-conservative flow counting is used to count how many Contributing Flows are represented in an Aggregated Flow. Flow counters are not distributed in this case. An Original Flow which is present within N Aggregated Flows would add N to the sum of non-conservative flow counts, one to each Aggregated Flow. In other words, the sum of conservative flow counts over a set of Aggregated Flows is always equal to the number of Original Flows, while the sum of non-conservative flow counts is strictly greater than or equal to the number of Original Flows.

For example, consider Flows A, B, and C as illustrated in Figure 5. Assume that the key aggregation step aggregates the keys of these three Flows to the same aggregated Flow Key, and that start interval counter distribution is in effect. The conservative flow count for interval 0 is 3 (since Flows A, B, and C all begin in this interval), and for the other two intervals is 0. The non-conservative flow count for interval 0 is also 3 (due to the presence of Flows A, B, and C), for interval 1 is 2 (Flows B and C), and for interval 2 is 1 (Flow C). The sum of the conservative counts $3 + 0 + 0 = 3$, the number of Original Flows; while the sum of the non-conservative counts $3 + 2 + 1 = 6$.

Note that the active and inactive timeouts used to generate Original Flows, as well as the cache policy used to generate those Flows, have an effect on how meaningful either the conservative or non-conservative flow count will be during aggregation. In general, all the Original Exporters producing Original Flows to be aggregated SHOULD be aggregated using caches configured identically or similarly. Original Exporters using the IPFIX Configuration Model

SHOULD be configured to export Flows with equal or similar activeTimeout and inactiveTimeout configuration values, and the same cacheMode, as defined in [I-D.ietf-ipfix-configuration-model].

5.2.2. Counting Distinct Key Values

One common case in aggregation is counting distinct key values that were reduced away during key aggregation. The most common use case for this is counting distinct hosts per Flow Key; for example, in host characterization or anomaly detection, distinct sources per destination or distinct destinations per source are common metrics. These new non-Key fields are added during key aggregation.

For such applications, Information Elements for distinct counts of IPv4 and IPv6 addresses are defined in Section 7.3. These are named distinctCountOf(KeyName). Additional such Information Elements SHOULD be registered with IANA on an as-needed basis.

5.3. Spatial Aggregation of Non-Key Fields

Aggregation operations may also lead to the addition of value fields demoted from key fields, or derived from other value fields in the Original Flows. Specific cases of this are treated in the subsections below.

5.3.1. Counter Statistics

Some applications of aggregation may benefit from computing different statistics than those native to each non-key field (i.e., union for flags, sum for counters). For example, minimum and maximum packet counts per Flow, mean bytes per packet per Contributing Flow, and so on. Certain Information Elements for these applications are already provided in the IANA IPFIX Information Elements registry (<http://www.iana.org/assignments/ipfix/ipfix.html> (e.g. minimumIpTotalLength)).

A complete specification of additional aggregate counter statistics is outside the scope of this document, and should be added in the future to the IANA IPFIX Information Elements registry on a per-application, as-needed basis.

5.3.2. Derivation of New Values from Flow Keys and non-Key fields

More complex operations may lead to other derived fields being generated from the set of values or Flow Keys reduced away during aggregation. A prime example of this is sample entropy calculation. This counts distinct values and frequency, so is similar to distinct key counting as in Section 5.2.2, but may be applied to the

distribution of values for any flow field.

Sample entropy calculation provides a one-number normalized representation of the value spread and is useful for anomaly detection. The behaviour of entropy statistics is such that a small number of keys showing up very often drives the entropy value down towards zero, while a large number of keys, each showing up with lower frequency drives the entropy value up.

Entropy statistics are generally useful for address-like keys, like IP addresses, port numbers, AS numbers, etc. They can also be done on flow length, flow duration fields and the like, even if this generally yields less distinct value shifts when the traffic mix changes.

As a practical example, one host scanning a lot of other hosts will drive source IP entropy down and target IP entropy up. A similar effect can be observed for ports. This pattern can also be caused by the scan-traffic of a fast Internet worm. A second example would be a DDoS flooding attack against a single target (or small number of targets) which drives source IP entropy up and target IP entropy down.

A complete specification of additional derived values or entropy information elements is outside the scope of this document. Any such Information Elements should be added in the future to the IANA IPFIX Information Elements registry on a per-application, as-needed basis. However, in the special case of entropy calculations, to support comparability of entropies of fields with different bit sizes, entropy SHOULD be represented as a float32 or float64 value normalized to the range [0..1].

5.4. Aggregation Combination

Interval distribution and key aggregation together may generate multiple Partially Aggregated Flows covering the same time interval with the same set of Flow Key values. The process of combining these Partially Aggregated Flows into a single Aggregated Flow is called aggregation combination. In general, non-Key values from multiple Contributing Flows are combined using the same operation by which values are combined from packets to form Flows for each Information Element. Counters are summed, averages are averaged, flags are unioned, and so on.

6. Additional Considerations and Special Cases in Flow Aggregation

6.1. Exact versus Approximate Counting during Aggregation

In certain circumstances, particularly involving aggregation by devices with limited resources, and in situations where exact aggregated counts are less important than relative magnitudes (e.g. driving graphical displays), counter distribution during key aggregation may be performed by approximate counting means (e.g. Bloom filters). The choice to use approximate counting is implementation- and application-dependent.

6.2. Delay and Loss introduced by the IAP

When accepting Original Flows in export order from traffic captured live, the Intermediate Aggregation Process wait for all Original Flows which may contribute to a given interval during interval distribution. This is generally dominated by the active timeout of the Metering Process measuring the Original Flows. For example, with Metering Processes configured with a 5 minute active timeout, the Intermediate Aggregation Process introduces a delay of at least 5 minutes to all exported Aggregated Flows to ensure it has received all Original Flows.

In certain circumstances, additional delay at the original Exporter may cause an IAP to close an interval before the last Original Flow(s) accountable to the interval arrives; in this case the IAP SHOULD drop the late Original Flow(s). Accounting of flows lost at an Intermediate Process due to such issues is covered in [I-D.ietf-ipfix-mediation-protocol].

6.3. Considerations for Aggregation of Sampled Flows

The accuracy of Aggregated Flows may also be affected by sampling of the Original Flows, or sampling of packets making up the Original Flows. At the time of writing, the effect of sampling on flow aggregation is still an open research question. However, to maximize the comparability of Aggregated Flows, aggregation of sampled Flows SHOULD only use Original Flows sampled using the same sampling rate and sampling algorithm, Flows created from packets sampled using the same sampling rate and sampling algorithm, or Original Flows which have been normalized as if they had the same sampling rate and algorithm before aggregation. For more on packet sampling within IPFIX, see [RFC5476]. For more on Flow sampling within the IPFIX Mediator Framework, see [I-D.ietf-ipfix-flow-selection-tech].

6.4. Considerations for Aggregation of Heterogeneous Flows

Aggregation may be applied to Original Flows from different sources and of different types (i.e., represented using different, perhaps

wildly-different Templates). When the goal is to separate the heterogeneous Original Flows and aggregate them into heterogeneous Aggregated Flows, each aggregation should be done at its own Intermediate Aggregation Process. The Observation Domain ID on the Messages containing the output Aggregated Flows can be used to identify the different Processes, and to segregate the output.

However, when the goal is to aggregate these Flows into a single stream of Aggregated Flows representing one type of data, and if the Original Flows may represent the same original packet at two different Observation Points, the Original Flows should be correlated by the correlation and normalization operation within the IAP to ensure that each packet is only represented in a single Aggregated Flow or set of Aggregated Flows differing only by aggregation interval.

7. Export of Aggregated IP Flows using IPFIX

In general, Aggregated Flows are exported in IPFIX as any normal Flow. However, certain aspects of Aggregated Flow export benefit from additional guidelines, or new Information Elements to represent aggregation metadata or information generated during aggregation. These are detailed in the following subsections.

7.1. Time Interval Export

Since an Aggregated Flow is simply a Flow, the existing timestamp Information Elements in the IPFIX Information Model (e.g., flowStartMilliseconds, flowEndNanoseconds) are sufficient to specify the time interval for aggregation. Therefore, this document specifies no new aggregation-specific Information Elements for exporting time interval information.

Each Aggregated Flow carrying timing information SHOULD contain both an interval start and interval end timestamp. If an exporter of Aggregated Flows omits the interval end timestamp from each Aggregated Flow, the time interval for Aggregated Flows within an Observation Domain and Transport Session MUST be regular and constant. However, note that this approach might lead to interoperability problems when exporting Aggregated Flows to non-aggregation-aware Collecting Processes and downstream analysis tasks; therefore, an Exporting Process capable of exporting only interval start timestamps MUST provide a configuration option to export interval end timestamps as well.

7.2. Flow Count Export

The following four Information Elements are defined to count Original Flows as discussed in Section 5.2.1.

7.2.1. originalFlowsPresent

Description: The non-conservative count of Original Flows contributing to this Aggregated Flow. Non-conservative counts need not sum to the original count on re-aggregation.

Abstract Data Type: unsigned64

ElementId: TBD1

Status: Current

7.2.2. originalFlowsInitiated

Description: The conservative count of Original Flows whose first packet is represented within this Aggregated Flow. Conservative counts must sum to the original count on re-aggregation.

Abstract Data Type: unsigned64

ElementId: TBD2

Status: Current

7.2.3. originalFlowsCompleted

Description: The conservative count of Original Flows whose last packet is represented within this Aggregated Flow. Conservative counts must sum to the original count on re-aggregation.

Abstract Data Type: unsigned64

ElementId: TBD3

Status: Current

7.2.4. deltaFlowCount

Description: The conservative count of Original Flows contributing to this Aggregated Flow; may be distributed via any of the methods described in Section 5.1.1. This Information Element is compatible with Information Element 3 as used in NetFlow version 9.

Abstract Data Type: unsigned64

ElementId: 3

Status: Current

7.3. Distinct Host Export

The following four Information Elements represent the distinct counts of source and destination network-layer addresses, used to export distinct host counts reduced away during key aggregation.

7.3.1. distinctCountOfSourceIPAddress

Description: The count of distinct source IP address values for Original Flows contributing to this Aggregated Flow, without regard to version. This Information Element is preferred to the IP-version-specific counters, unless it is important to separate the counts by version.

Abstract Data Type: unsigned64

ElementId: TBD4

Status: Current

7.3.2. distinctCountOfDestinationIPAddress

Description: The count of distinct destination IP address values for Original Flows contributing to this Aggregated Flow, without regard to version. This Information Element is preferred to the version-specific counters below, unless it is important to separate the counts by version.

Abstract Data Type: unsigned64

ElementId: TBD5

Status: Current

7.3.3. distinctCountOfSourceIPv4Address

Description: The count of distinct source IPv4 address values for Original Flows contributing to this Aggregated Flow.

Abstract Data Type: unsigned32

ElementId: TBD6

Status: Current

7.3.4. distinctCountOfDestinationIPv4Address

Description: The count of distinct destination IPv4 address values for Original Flows contributing to this Aggregated Flow.

Abstract Data Type: unsigned32

ElementId: TBD7

Status: Current

7.3.5. distinctCountOfSourceIPv6Address

Description: The count of distinct source IPv6 address values for Original Flows contributing to this Aggregated Flow.

Abstract Data Type: unsigned64

ElementId: TBD8

Status: Current

7.3.6. distinctCountOfDestinationIPv6Address

Description: The count of distinct destination IPv6 address values for Original Flows contributing to this Aggregated Flow.

Abstract Data Type: unsigned64

ElementId: TBD9

Status: Current

7.4. Aggregate Counter Distribution Export

When exporting counters distributed among Aggregated Flows, as described in Section 5.1.1, the Exporting Process MAY export an Aggregate Counter Distribution Option Record for each Template describing Aggregated Flow records; this Options Template is described below. It uses the valueDistributionMethod Information Element, also defined below. Since in many cases distribution is simple, accounting the counters from Contributing Flows to the first

Interval to which they contribute, this is the default situation, for which no Aggregate Counter Distribution Record is necessary; Aggregate Counter Distribution Records are only applicable in more exotic situations, such as using an Aggregation Interval smaller than the durations of Original Flows.

7.4.1. Aggregate Counter Distribution Options Template

This Options Template defines the Aggregate Counter Distribution Record, which allows the binding of a value distribution method to a Template ID. Note that this Options Template causes the valueDistributionMethod to be implicitly scoped to the Observation Domain ID of the IPFIX Message containing the Aggregate Counter Distribution Record. This is used to signal to the Collecting Process how the counters were distributed. The fields are as below:

IE	Description
templateId [scope]	The Template ID of the Template defining the Aggregated Flows to which this distribution option applies. This Information Element MUST be defined as a Scope Field.
valueDistributionMethod	The method used to distribute the counters for the Aggregated Flows defined by the associated Template.

7.4.2. valueDistributionMethod Information Element

Description: A description of the method used to distribute the counters from Contributing Flows into the Aggregated Flow records described by an associated scope, generally a Template. The method is deemed to apply to all the non-key Information Elements in the referenced scope for which value distribution is a valid operation; if the originalFlowsInitiated and/or originalFlowsCompleted Information Elements appear in the Template, they are not subject to this distribution method, as they each infer their own distribution method. This is intended to be a complete set of possible value distribution methods; it is taken from the discussion Section 5.1.1 and encoded as follows:

Value	Description
0	Arbitrary: The counters for an Original Flow are explicitly not distributed according to any defined method.
1	Start Interval: The counters for an Original Flow are added to the counters of the appropriate Aggregated Flow containing the start time of the Original Flow. This should be assumed the default if value distribution information is not available at a Collecting Process for an Aggregated Flow.
2	End Interval: The counters for an Original Flow are added to the counters of the appropriate Aggregated Flow containing the end time of the Original Flow.
3	Mid Interval: The counters for an Original Flow are added to the counters of a single appropriate Aggregated Flow containing some timestamp between start and end time of the Original Flow.
4	Simple Uniform Distribution: Each counter for an Original Flow is divided by the number of time intervals the Original Flow covers (i.e., of appropriate Aggregated Flows sharing the same Flow Key), and this number is added to each corresponding counter in each Aggregated Flow.
5	Proportional Uniform Distribution: Each counter for an Original Flow is divided by the number of time _units_ the Original Flow covers, to derive a mean count rate. This mean count rate is then multiplied by the number of time units in the intersection of the duration of the Original Flow and the time interval of each Aggregated Flow. This is like simple uniform distribution, but accounts for the fractional portions of a time interval covered by an Original Flow in the first and last time interval.

6	Simulated Process: Each counter of the Original Flow is distributed among the intervals of the Aggregated Flows according to some function the Aggregation Process uses based upon properties of Flows presumed to be like the Original Flow. This is essentially an assertion that the Aggregation Process has no direct packet timing information but is nevertheless not using one of the other simpler distribution methods. The Aggregation Process specifically makes no assertion as to the correctness of the simulation.
7	Direct: The Aggregation Process has access to the original packet timings from the packets making up the Original Flow, and uses these to distribute or recalculate the counters.

Abstract Data Type: unsigned8

ElementId: TBD10

Status: Current

8. Examples

In these examples, the same data, described by the same template, will be aggregated multiple different ways; this illustrates the various different functions which could be implemented by Intermediate Aggregation Processes. Templates are shown in IESpec format as introduced in [I-D.ietf-ipfix-ie-doctors]. The source data format is a simplified flow: timestamps, traditional 5-tuple, and octet count. The template is shown in Figure 8.

```
flowStartMilliseconds(152)[8]
flowEndMilliseconds(153)[8]
sourceIPv4Address(8)[4]
destinationIPv4Address(12)[4]
sourceTransportPort(7)[2]
destinationTransportPort(11)[2]
protocolIdentifier(4)[1]
octetDeltaCount(1)[8]
```

Figure 8: Input template for examples

The data records given as input to the examples in this section are shown below, in the format "flowStartMilliseconds-flowEndMilliseconds sourceIPv4Address:sourceTransportPort -> destinationIPv4Address:

destinationTransportPort (protocolIdentifier) octetDeltaCount";
 timestamps are given in H:MM:SS.sss format.

start time	end time	source ip4	port	dest ip4	port	pt	oct
9:00:00.138	9:00:00.138	192.0.2.2	47113	192.0.2.131	53	17	119
9:00:03.246	9:00:03.246	192.0.2.2	22153	192.0.2.131	53	17	83
9:00:00.478	9:00:03.486	192.0.2.2	52420	198.51.100.2	443	6	1637
9:00:07.172	9:00:07.172	192.0.2.3	56047	192.0.2.131	53	17	111
9:00:07.309	9:00:14.861	192.0.2.3	41183	198.51.100.67	80	6	16838
9:00:03.556	9:00:19.876	192.0.2.2	17606	198.51.100.68	80	6	11538
9:00:25.210	9:00:25.210	192.0.2.3	47113	192.0.2.131	53	17	119
9:00:26.358	9:00:30.198	192.0.2.3	48458	198.51.100.133	80	6	2973
9:00:29.213	9:01:00.061	192.0.2.4	61295	198.51.100.2	443	6	8350
9:04:00.207	9:04:04.431	203.0.113.3	41256	198.51.100.133	80	6	778
9:03:59.624	9:04:06.984	203.0.113.3	51662	198.51.100.3	80	6	883
9:00:30.532	9:06:15.402	192.0.2.2	37581	198.51.100.2	80	6	15420
9:06:56.813	9:06:59.821	203.0.113.3	52572	198.51.100.2	443	6	1637
9:06:30.565	9:07:00.261	203.0.113.3	49914	197.51.100.133	80	6	561
9:06:55.160	9:07:05.208	192.0.2.2	50824	198.51.100.2	443	6	1899
9:06:49.322	9:07:05.322	192.0.2.3	34597	198.51.100.3	80	6	1284
9:07:05.849	9:07:09.625	203.0.113.3	58907	198.51.100.4	80	6	2670
9:10:45.161	9:10:45.161	192.0.2.4	22478	192.0.2.131	53	17	75
9:10:45.209	9:11:01.465	192.0.2.4	49513	198.51.100.68	80	6	3374
9:10:57.094	9:11:00.614	192.0.2.4	64832	198.51.100.67	80	6	138
9:10:59.770	9:11:02.842	192.0.2.3	60833	198.51.100.69	443	6	2325
9:02:18.390	9:13:46.598	203.0.113.3	39586	198.51.100.17	80	6	11200
9:13:53.933	9:14:06.605	192.0.2.2	19638	198.51.100.3	80	6	2869
9:13:02.864	9:14:08.720	192.0.2.3	40429	198.51.100.4	80	6	18289

Figure 9: Input data for examples

8.1. Traffic Time-Series per Source

Aggregating flows by source IP address in time series (i.e., with a regular interval) can be used in subsequent heavy-hitter analysis and as a source parameter for statistical anomaly detection techniques. Here, the Intermediate Aggregation Process imposes an interval, aggregates the key to remove all key fields other than the source IP address, then combines the result into a stream of Aggregated Flows. The imposed interval of 5 minutes is longer than the majority of flows; for those flows crossing interval boundaries, the entire flow is accounted to the interval containing the start time of the flow.

In this example the Partially Aggregated Flows after each conceptual operation in the Intermediate Aggregation Process are shown. These are meant to be illustrative of the conceptual operations only, and not to suggest an implementation (indeed, the example shown here would not necessarily be the most efficient method for performing

these operations). Subsequent examples will omit the Partially Aggregated Flows for brevity.

The input to this process could be any Flow Record containing a source IP address and octet counter; consider for this example the template and data from the introduction. The Intermediate Aggregation Process would then output records containing just timestamps, source IP, and octetDeltaCount, as in Figure 10.

```
flowStartMilliseconds(152)[8]
flowEndMilliseconds(153)[8]
sourceIPv4Address(8)[4]
octetDeltaCount(1)[8]
```

Figure 10: Output template for time series per source

Assume the goal is to get 5-minute (300s) time series of octet counts per source IP address. The aggregation operations would then be arranged as in Figure 11.

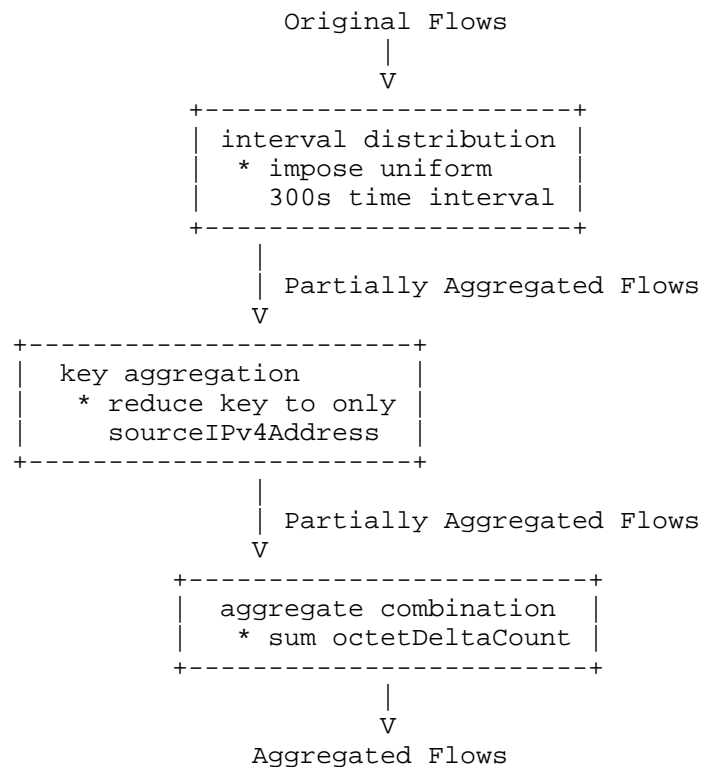


Figure 11: Aggregation operations for time series per source

After the interval distribution step, only the time intervals have changed; the Partially Aggregated flows are shown in Figure 12. Note that interval distribution follows the default Start Interval policy; that is, the entire flow is accounted to the interval containing the flow's start time.

start time	end time	source ip4	port	dest ip4	port	pt	oct
9:00:00.000	9:05:00.000	192.0.2.2	47113	192.0.2.131	53	17	119
9:00:00.000	9:05:00.000	192.0.2.2	22153	192.0.2.131	53	17	83
9:00:00.000	9:05:00.000	192.0.2.2	52420	198.51.100.2	443	6	1637
9:00:00.000	9:05:00.000	192.0.2.3	56047	192.0.2.131	53	17	111
9:00:00.000	9:05:00.000	192.0.2.3	41183	198.51.100.67	80	6	16838
9:00:00.000	9:05:00.000	192.0.2.2	17606	198.51.100.68	80	6	11538
9:00:00.000	9:05:00.000	192.0.2.3	47113	192.0.2.131	53	17	119
9:00:00.000	9:05:00.000	192.0.2.3	48458	198.51.100.133	80	6	2973
9:00:00.000	9:05:00.000	192.0.2.4	61295	198.51.100.2	443	6	8350
9:00:00.000	9:05:00.000	203.0.113.3	41256	198.51.100.133	80	6	778
9:00:00.000	9:05:00.000	203.0.113.3	51662	198.51.100.3	80	6	883
9:00:00.000	9:05:00.000	192.0.2.2	37581	198.51.100.2	80	6	15420
9:00:00.000	9:05:00.000	203.0.113.3	39586	198.51.100.17	80	6	11200
9:05:00.000	9:10:00.000	203.0.113.3	52572	198.51.100.2	443	6	1637
9:05:00.000	9:10:00.000	203.0.113.3	49914	197.51.100.133	80	6	561
9:05:00.000	9:10:00.000	192.0.2.2	50824	198.51.100.2	443	6	1899
9:05:00.000	9:10:00.000	192.0.2.3	34597	198.51.100.3	80	6	1284
9:05:00.000	9:10:00.000	203.0.113.3	58907	198.51.100.4	80	6	2670
9:10:00.000	9:15:00.000	192.0.2.4	22478	192.0.2.131	53	17	75
9:10:00.000	9:15:00.000	192.0.2.4	49513	198.51.100.68	80	6	3374
9:10:00.000	9:15:00.000	192.0.2.4	64832	198.51.100.67	80	6	138
9:10:00.000	9:15:00.000	192.0.2.3	60833	198.51.100.69	443	6	2325
9:10:00.000	9:15:00.000	192.0.2.2	19638	198.51.100.3	80	6	2869
9:10:00.000	9:15:00.000	192.0.2.3	40429	198.51.100.4	80	6	18289

Figure 12: Interval imposition for time series per source

After the key aggregation step, all Flow Keys except the source IP address have been discarded, as shown in Figure 13. This leaves duplicate Partially Aggregated flows to be combined in the final operation.

start time	end time	source ip4	octets
9:00:00.000	9:05:00.000	192.0.2.2	119
9:00:00.000	9:05:00.000	192.0.2.2	83
9:00:00.000	9:05:00.000	192.0.2.2	1637
9:00:00.000	9:05:00.000	192.0.2.3	111
9:00:00.000	9:05:00.000	192.0.2.3	16838
9:00:00.000	9:05:00.000	192.0.2.2	11538
9:00:00.000	9:05:00.000	192.0.2.3	119
9:00:00.000	9:05:00.000	192.0.2.3	2973
9:00:00.000	9:05:00.000	192.0.2.4	8350
9:00:00.000	9:05:00.000	203.0.113.3	778
9:00:00.000	9:05:00.000	203.0.113.3	883
9:05:00.000	9:10:00.000	203.0.113.3	1637
9:05:00.000	9:10:00.000	203.0.113.3	561
9:05:00.000	9:10:00.000	192.0.2.2	1899
9:05:00.000	9:10:00.000	192.0.2.3	1284
9:05:00.000	9:10:00.000	203.0.113.3	2670
9:10:00.000	9:15:00.000	192.0.2.4	75
9:10:00.000	9:15:00.000	192.0.2.4	3374
9:10:00.000	9:15:00.000	192.0.2.4	138
9:10:00.000	9:15:00.000	192.0.2.3	2325
9:10:00.000	9:15:00.000	192.0.2.2	2869
9:10:00.000	9:15:00.000	192.0.2.3	18289

Figure 13: Key aggregation for time series per source

Aggregate combination sums the counters per key and interval; the summations of the first two keys and intervals are shown in detail in Figure 14.

start time	end time	source ip4	octets	
9:00:00.000	9:05:00.000	192.0.2.2	119	
9:00:00.000	9:05:00.000	192.0.2.2	83	
9:00:00.000	9:05:00.000	192.0.2.2	1637	
9:00:00.000	9:05:00.000	192.0.2.2	11538	
+	9:00:00.000	9:05:00.000	192.0.2.2	15420

=	9:00:00.000	9:05:00.000	192.0.2.2	28797
9:00:00.000	9:05:00.000	192.0.2.3	111	
9:00:00.000	9:05:00.000	192.0.2.3	16838	
9:00:00.000	9:05:00.000	192.0.2.3	119	
+	9:00:00.000	9:05:00.000	192.0.2.3	2973

=	9:00:00.000	9:05:00.000	192.0.2.3	20041

Figure 14: Summation during aggregate combination

Applying this to each set of Partially Aggregated Flows to produce the final Aggregated Flows shown in Figure 15 to be exported by the template in Figure 10.

start time	end time	source ip4	octets
9:00:00.000	9:05:00.000	192.0.2.2	28797
9:00:00.000	9:05:00.000	192.0.2.3	20041
9:00:00.000	9:05:00.000	192.0.2.4	8350
9:00:00.000	9:05:00.000	203.0.113.3	12861
9:05:00.000	9:10:00.000	192.0.2.2	1899
9:05:00.000	9:10:00.000	192.0.2.3	1284
9:05:00.000	9:10:00.000	203.0.113.3	4868
9:10:00.000	9:15:00.000	192.0.2.2	2869
9:10:00.000	9:15:00.000	192.0.2.3	20614
9:10:00.000	9:15:00.000	192.0.2.4	3587

Figure 15: Aggregated Flows for time series per source

8.2. Core Traffic Matrix

Aggregating flows by source and destination autonomous system number in time series is used to generate core traffic matrices. The core traffic matrix provides a view of the state of the routes within a network, and can be used for long-term planning of changes to network design based on traffic demand. Here, imposed time intervals are generally much longer than active flow timeouts. The traffic matrix is reported in terms of octets, packets, and flows, as each of these values may have a subtly different effect on capacity planning.

This example demonstrates key aggregation using derived keys and original flow counting. While some Original Flows may be generated by Exporting Processes on forwarding devices, and therefore contain the `bgpSourceAsNumber` and `bgpDestinationAsNumber` Information Elements, Original Flows from Exporting Processes on dedicated measurement devices will contain only a `destinationIPv[46]Address`. For these flows, the Mediator must look up a next hop AS from a IP to AS table, replacing source and destination addresses with AS numbers. The table used in this example is shown in Figure 16. (Note that due to limited example address space, in this example we ignore the common practice of routing only blocks of /24 or larger).

prefix	ASN
192.0.2.0/25	64496
192.0.2.128/25	64497
198.51.100/24	64498
203.0.113.0/24	64499

Figure 16: Example Autonomous system number map

The template for Aggregated Flows produced by this example is shown in Figure 17.

```
flowStartMilliseconds(152)[8]  
flowEndMilliseconds(153)[8]  
bgpSourceAsNumber(16)[4]  
bgpDestinationAsNumber(17)[4]  
octetDeltaCount(1)[8]
```

Figure 17: Output template for traffic matrix

Assume the goal is to get 60-minute time series of octet counts per source/destination ASN pair. The aggregation operations would then be arranged as in Figure 18.

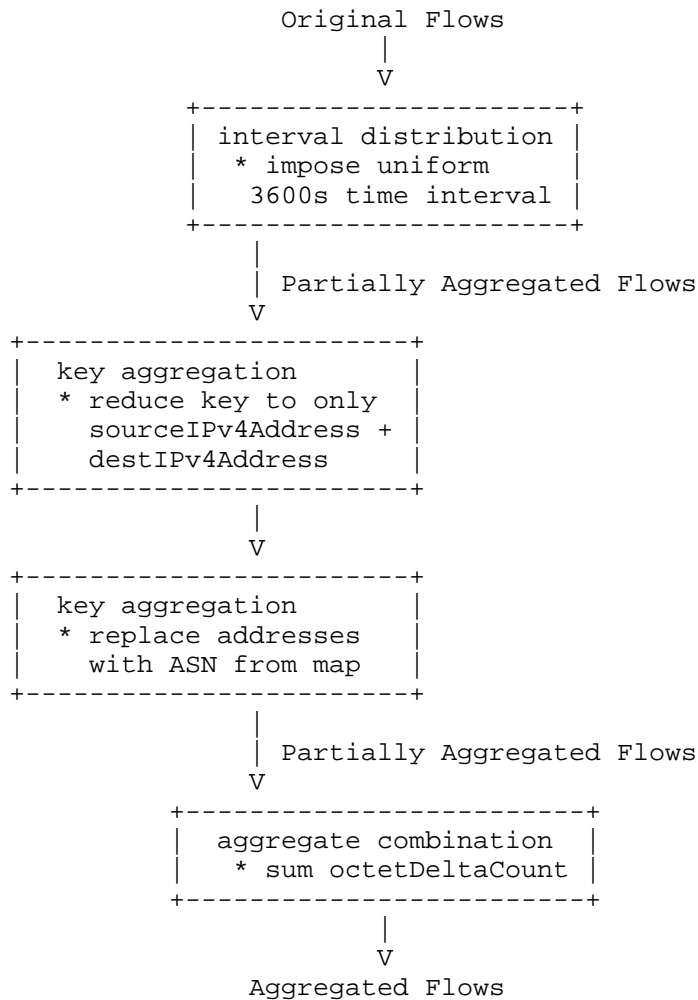


Figure 18: Aggregation operations for traffic matrix

After the interval distribution step, only the time intervals have changed; the Partially Aggregated flows are shown in Figure 19. Note that the flows are identical to those in interval distribution step in the previous example, except the chosen interval (1 hour, 3600 seconds) is different; therefore, all the flows fit into a single interval.

start time	end time	source ip4	port	dest ip4	port	pt	oct
9:00:00	10:00:00	192.0.2.2	47113	192.0.2.131	53	17	119
9:00:00	10:00:00	192.0.2.2	22153	192.0.2.131	53	17	83
9:00:00	10:00:00	192.0.2.2	52420	198.51.100.2	443	6	1637
9:00:00	10:00:00	192.0.2.3	56047	192.0.2.131	53	17	111
9:00:00	10:00:00	192.0.2.3	41183	198.51.100.67	80	6	16838
9:00:00	10:00:00	192.0.2.2	17606	198.51.100.68	80	6	11538
9:00:00	10:00:00	192.0.2.3	47113	192.0.2.131	53	17	119
9:00:00	10:00:00	192.0.2.3	48458	198.51.100.133	80	6	2973
9:00:00	10:00:00	192.0.2.4	61295	198.51.100.2	443	6	8350
9:00:00	10:00:00	203.0.113.3	41256	198.51.100.133	80	6	778
9:00:00	10:00:00	203.0.113.3	51662	198.51.100.3	80	6	883
9:00:00	10:00:00	192.0.2.2	37581	198.51.100.2	80	6	15420
9:00:00	10:00:00	203.0.113.3	52572	198.51.100.2	443	6	1637
9:00:00	10:00:00	203.0.113.3	49914	197.51.100.133	80	6	561
9:00:00	10:00:00	192.0.2.2	50824	198.51.100.2	443	6	1899
9:00:00	10:00:00	192.0.2.3	34597	198.51.100.3	80	6	1284
9:00:00	10:00:00	203.0.113.3	58907	198.51.100.4	80	6	2670
9:00:00	10:00:00	192.0.2.4	22478	192.0.2.131	53	17	75
9:00:00	10:00:00	192.0.2.4	49513	198.51.100.68	80	6	3374
9:00:00	10:00:00	192.0.2.4	64832	198.51.100.67	80	6	138
9:00:00	10:00:00	192.0.2.3	60833	198.51.100.69	443	6	2325
9:00:00	10:00:00	203.0.113.3	39586	198.51.100.17	80	6	11200
9:00:00	10:00:00	192.0.2.2	19638	198.51.100.3	80	6	2869
9:00:00	10:00:00	192.0.2.3	40429	198.51.100.4	80	6	18289

Figure 19: Interval imposition for traffic matrix

The next step is to discard irrelevant key fields, and replace the source and destination addresses with source and destination AS numbers in the map; the results of these key aggregation steps are shown in Figure 20.

start time	end time	source ASN	dest ASN	octets
9:00:00	10:00:00	AS64496	AS64497	119
9:00:00	10:00:00	AS64496	AS64497	83
9:00:00	10:00:00	AS64496	AS64498	1637
9:00:00	10:00:00	AS64496	AS64497	111
9:00:00	10:00:00	AS64496	AS64498	16838
9:00:00	10:00:00	AS64496	AS64498	11538
9:00:00	10:00:00	AS64496	AS64497	119
9:00:00	10:00:00	AS64496	AS64498	2973
9:00:00	10:00:00	AS64496	AS64498	8350
9:00:00	10:00:00	AS64499	AS64498	778
9:00:00	10:00:00	AS64499	AS64498	883
9:00:00	10:00:00	AS64496	AS64498	15420
9:00:00	10:00:00	AS64499	AS64498	1637
9:00:00	10:00:00	AS64499	AS64498	561
9:00:00	10:00:00	AS64496	AS64498	1899
9:00:00	10:00:00	AS64496	AS64498	1284
9:00:00	10:00:00	AS64499	AS64498	2670
9:00:00	10:00:00	AS64496	AS64497	75
9:00:00	10:00:00	AS64496	AS64498	3374
9:00:00	10:00:00	AS64496	AS64498	138
9:00:00	10:00:00	AS64496	AS64498	2325
9:00:00	10:00:00	AS64499	AS64498	11200
9:00:00	10:00:00	AS64496	AS64498	2869
9:00:00	10:00:00	AS64496	AS64498	18289

Figure 20: Key aggregation for traffic matrix: reduction and replacement

Finally, aggregate combination sums the counters per key and interval. The resulting Aggregated Flows containing the traffic matrix, shown in Figure 21, are then exported using the template in Figure 17. Note that these aggregated flows represent a sparse matrix: AS pairs for which no traffic was received have no corresponding record in the output.

start time	end time	source ASN	dest ASN	octets
9:00:00	10:00:00	AS64496	AS64497	507
9:00:00	10:00:00	AS64496	AS64498	86934
9:00:00	10:00:00	AS64499	AS64498	17729

Figure 21: Aggregated Flows for traffic matrix

The output of this operation is suitable for re-aggregation: that is, traffic matrices from single links or observation points can be aggregated through the same interval imposition and aggregate combination steps in order to build a traffic matrix for an entire network.

8.3. Distinct Source Count per Destination Endpoint

Aggregating flows by destination address and port, and counting distinct sources aggregated away, can be used as part of passive service inventory and host characterization approaches. This example shows aggregation as an analysis technique, performed on source data stored in an IPFIX File. As the Transport Session in this File is bounded, removal of all timestamp information allows summarization of the entire time interval contained within the interval. Removal of timing information during interval imposition is equivalent to an infinitely long imposed time interval. This demonstrates both how infinite intervals work, and how unique counters work. The aggregation operations are summarized in Figure 22.

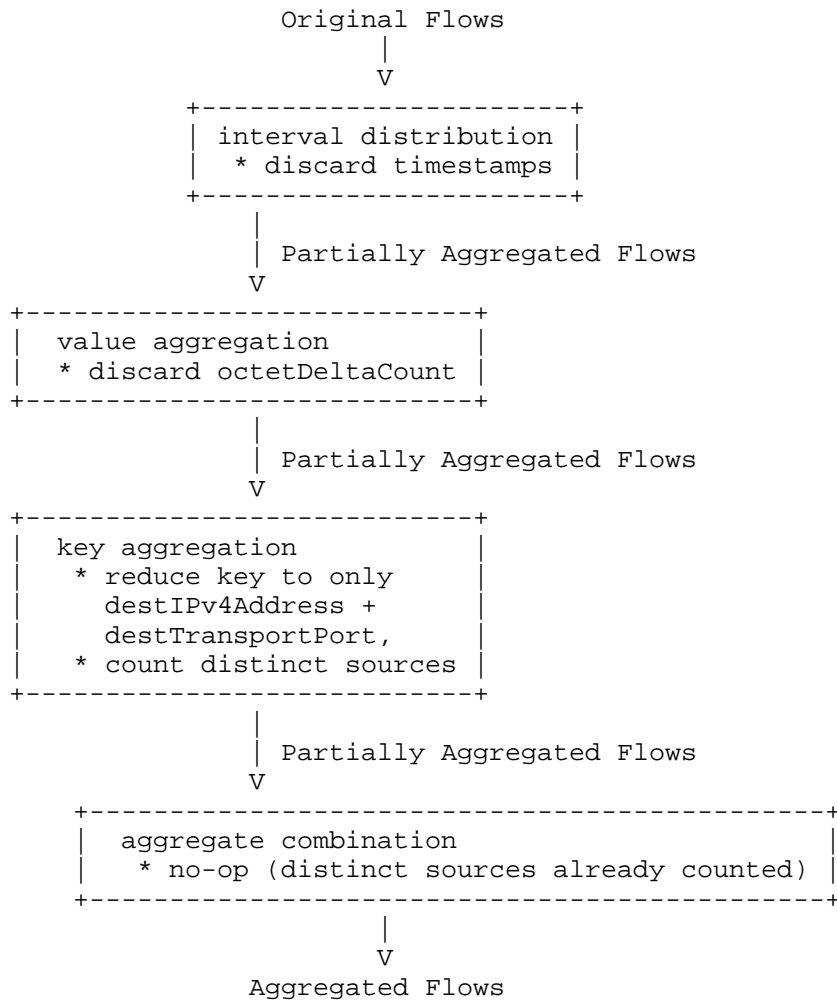


Figure 22: Aggregation operations for source count

The template for Aggregated Flows produced by this example is shown in Figure 23.

```

destinationIPv4Address(12)[4]
destinationTransportPort(11)[2]
distinctCountOfSourceIPAddress(TBD4)[8]

```

Figure 23: Output template for source count

Interval distribution, in this case, merely discards the timestamp information from the Original Flows, and as such is not shown.

Likewise, the value aggregation step simply discards the `octetDeltaCount` value field. The key aggregation step reduces the key to the `destinationIPv4Address` and `destinationTransportPort`, counting the distinct source addresses. Since this is essentially the output of this aggregation function, the aggregate combination operation is a no-op; the resulting Aggregated Flows are shown in Figure 24.

dest ip4	port	dist src
192.0.2.131	53	3
198.51.100.2	80	1
198.51.100.2	443	3
198.51.100.67	80	2
198.51.100.68	80	2
198.51.100.133	80	2
198.51.100.3	80	3
198.51.100.4	80	2
198.51.100.17	80	1
198.51.100.69	443	1

Figure 24: Aggregated flows for source count

8.4. Traffic Time-Series per Source with Counter Distribution

Returning to the example in Section 8.1, note that our source data contains some flows with durations longer than the imposed interval of five minutes. The default method for dealing with such flows is to account them to the interval containing the flow's start time.

In this example, the same data is aggregated using the same arrangement of operations and the same output template as the as in Section 8.1, but using a different counter distribution policy, Simple Uniform Distribution, as described in Section 5.1.1. In order to do this, the Exporting Process first exports the Aggregate Counter Distribution Options Template, as in Figure 25.

```
templateId(12)[2]{scope}
valueDistribtutionMethod(TBD10)[1]
```

Figure 25: Aggregate Counter Distribution Options Template

This is followed by an Aggregate Counter Distribution Record described by this Template; assuming the output template in Figure 10 has ID 257, this would appear as in Figure 26.

```
templateId 257: valueDistributionMethod 4 (Simple Uniform)
```

Figure 26: Aggregate Counter Distribution Record

Following metadata export, the aggregation steps follow as before. However, two long flows are distributed across multiple intervals in the interval imposition step, as indicated with "*" in Figure 27. Note the uneven distribution of the three-interval, 11200-octet flow into three Partially Aggregated Flows of 3733, 3733, and 3734 octets; this ensures no cumulative error is injected by the interval distribution step.

start time	end time	source ip4	port	dest ip4	port	pt	oct
9:00:00.000	9:05:00.000	192.0.2.2	47113	192.0.2.131	53	17	119
9:00:00.000	9:05:00.000	192.0.2.2	22153	192.0.2.131	53	17	83
9:00:00.000	9:05:00.000	192.0.2.2	52420	198.51.100.2	443	6	1637
9:00:00.000	9:05:00.000	192.0.2.3	56047	192.0.2.131	53	17	111
9:00:00.000	9:05:00.000	192.0.2.3	41183	198.51.100.67	80	6	16838
9:00:00.000	9:05:00.000	192.0.2.2	17606	198.51.100.68	80	6	11538
9:00:00.000	9:05:00.000	192.0.2.3	47113	192.0.2.131	53	17	119
9:00:00.000	9:05:00.000	192.0.2.3	48458	198.51.100.133	80	6	2973
9:00:00.000	9:05:00.000	192.0.2.4	61295	198.51.100.2	443	6	8350
9:00:00.000	9:05:00.000	203.0.113.3	41256	198.51.100.133	80	6	778
9:00:00.000	9:05:00.000	203.0.113.3	51662	198.51.100.3	80	6	883
9:00:00.000	9:05:00.000	192.0.2.2	37581	198.51.100.2	80	6	7710*
9:00:00.000	9:05:00.000	203.0.113.3	39586	198.51.100.17	80	6	3733*
9:05:00.000	9:10:00.000	203.0.113.3	52572	198.51.100.2	443	6	1637
9:05:00.000	9:10:00.000	203.0.113.3	49914	197.51.100.133	80	6	561
9:05:00.000	9:10:00.000	192.0.2.2	50824	198.51.100.2	443	6	1899
9:05:00.000	9:10:00.000	192.0.2.3	34597	198.51.100.3	80	6	1284
9:05:00.000	9:10:00.000	203.0.113.3	58907	198.51.100.4	80	6	2670
9:05:00.000	9:10:00.000	192.0.2.2	37581	198.51.100.2	80	6	7710*
9:05:00.000	9:10:00.000	203.0.113.3	39586	198.51.100.17	80	6	3733*
9:10:00.000	9:15:00.000	192.0.2.4	22478	192.0.2.131	53	17	75
9:10:00.000	9:15:00.000	192.0.2.4	49513	198.51.100.68	80	6	3374
9:10:00.000	9:15:00.000	192.0.2.4	64832	198.51.100.67	80	6	138
9:10:00.000	9:15:00.000	192.0.2.3	60833	198.51.100.69	443	6	2325
9:10:00.000	9:15:00.000	192.0.2.2	19638	198.51.100.3	80	6	2869
9:10:00.000	9:15:00.000	192.0.2.3	40429	198.51.100.4	80	6	18289
9:10:00.000	9:15:00.000	203.0.113.3	39586	198.51.100.17	80	6	3734*

Figure 27: Distirbuted interval imposition for time series per source

Subsequent steps are as in Section 8.1; the results, to be exported using Figure 10, are shown in Figure 28, with Aggregated Flows differing from the example in Section 8.1 indicated by "*".

start time	end time	source ip4	octets
9:00:00.000	9:05:00.000	192.0.2.2	21087*
9:00:00.000	9:05:00.000	192.0.2.3	20041
9:00:00.000	9:05:00.000	192.0.2.4	8350
9:00:00.000	9:05:00.000	203.0.113.3	9394*
9:05:00.000	9:10:00.000	192.0.2.2	9609*
9:05:00.000	9:10:00.000	192.0.2.3	1284
9:05:00.000	9:10:00.000	203.0.113.3	8601*
9:10:00.000	9:15:00.000	192.0.2.2	2869
9:10:00.000	9:15:00.000	192.0.2.3	20594
9:10:00.000	9:15:00.000	192.0.2.4	3587
9:10:00.000	9:15:00.000	203.0.113.3	3734*

Figure 28: Aggregated Flows for time series per source with counter distribution

9. Security Considerations

This document specifies the operation of an Intermediate Aggregation Process with the IPFIX Protocol; the Security Considerations for the protocol itself in Section 11 [RFC-EDITOR NOTE: verify section number] of [I-D.ietf-ipfix-protocol-rfc5101bis] therefore apply. In the common case that aggregation is performed on a Mediator, the Security Considerations for Mediators in Section 9 of [RFC6183] apply as well.

As mentioned in Section 3, certain aggregation operations may tend to have an anonymizing effect on flow data by obliterating sensitive identifiers. Aggregation may also be combined with anonymization within a Mediator, or as part of a chain of Mediators, to further leverage this effect. In any case in which an Intermediate Aggregation Process is applied as part of a data anonymization or protection scheme, or is used together with anonymization as described in [RFC6235], the Security Considerations in Section 9 of [RFC6235] apply.

10. IANA Considerations

This document specifies the creation of new IPFIX Information Elements in the IPFIX Information Element registry located at <http://www.iana.org/assignments/ipfix>, as defined in Section 7 above. IANA has assigned Information Element numbers to these Information Elements, and entered them into the registry.

[NOTE for IANA: The text TBDn should be replaced with the respective assigned Information Element numbers where they appear in this

document. Note that the deltaFlowCount Information Element has been assigned the number 3, as it is compatible with the corresponding existing (reserved) NetFlow v9 Information Element. Other Information Element numbers should be assigned outside the NetFlow V9 compatibility range, as these Information Elements are not supported by NetFlow V9.]

11. Acknowledgments

Special thanks to Elisa Boschi for early work on the concepts laid out in this document. Thanks to Paul Aitken, Lothar Braun, Christian Henke, and Rahul Patel for their reviews and valuable feedback. This work is materially supported by the European Union Seventh Framework Programme under grant agreement 257315 (DEMONS).

12. References

12.1. Normative References

- [I-D.ietf-ipfix-protocol-rfc5101bis]
Claise, B. and B. Trammell, "Specification of the IP Flow Information eXport (IPFIX) Protocol for the Exchange of Flow Information", draft-ietf-ipfix-protocol-rfc5101bis-02 (work in progress), June 2012.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

12.2. Informative References

- [RFC3917] Quittek, J., Zseby, T., Claise, B., and S. Zander, "Requirements for IP Flow Information Export (IPFIX)", RFC 3917, October 2004.
- [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.
- [RFC5476] Claise, B., Johnson, A., and J. Quittek, "Packet Sampling (PSAMP) Protocol Specifications", RFC 5476, March 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.
- [RFC5982] Kobayashi, A. and B. Claise, "IP Flow Information Export (IPFIX) Mediation: Problem Statement", RFC 5982, August 2010.
- [RFC6183] Kobayashi, A., Claise, B., Muenz, G., and K. Ishibashi, "IP Flow Information Export (IPFIX) Mediation: Framework", RFC 6183, April 2011.
- [RFC6235] Boschi, E. and B. Trammell, "IP Flow Anonymization Support", RFC 6235, May 2011.
- [I-D.ietf-ipfix-mediation-protocol]
Claise, B., Kobayashi, A., and B. Trammell, "Operation of the IP Flow Information Export (IPFIX) Protocol on IPFIX Mediators", draft-ietf-ipfix-mediation-protocol-01 (work in progress), June 2012.
- [I-D.ietf-ipfix-ie-doctors]
Trammell, B. and B. Claise, "Guidelines for Authors and Reviewers of IPFIX Information Elements", draft-ietf-ipfix-ie-doctors-03 (work in progress), June 2012.
- [I-D.ietf-ipfix-configuration-model]
Muenz, G., Claise, B., and P. Aitken, "Configuration Data Model for IPFIX and PSAMP", draft-ietf-ipfix-configuration-model-11 (work in progress), June 2012.
- [I-D.ietf-ipfix-flow-selection-tech]
D'Antonio, S., Zseby, T., Henke, C., and L. Peluso, "Flow Selection Techniques", draft-ietf-ipfix-flow-selection-tech-11 (work in progress), April 2012.
- [iana-ipfix-assignments]
Internet Assigned Numbers Authority, "IP Flow Information Export Information Elements (<http://www.iana.org/assignments/ipfix/ipfix.xml>)".

Authors' Addresses

Brian Trammell
Swiss Federal Institute of Technology Zurich
Gloriastrasse 35
8092 Zurich
Switzerland

Phone: +41 44 632 70 13
Email: trammell@tik.ee.ethz.ch

Arno Wagner
Consecom AG
Bleicherweg 64a
8002 Zurich
Switzerland

Email: arno@wagner.name

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

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

IP Flow Information Export WG
Internet-Draft
Intended status: Standards Track
Expires: January 8, 2012

G. Muenz
TU Muenchen
B. Claise
P. Aitken
Cisco Systems, Inc.
July 7, 2011

Configuration Data Model for IPFIX and PSAMP
<draft-ietf-ipfix-configuration-model-10>

Abstract

This document specifies a data model for configuring and monitoring Selection Processes, Caches, Exporting Processes, and Collecting Processes of IPFIX and PSAMP compliant Monitoring Devices using the NETCONF protocol. 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 Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

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

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

This Internet-Draft will expire on January 8, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	5
1.1. IPFIX Documents Overview	6
1.2. PSAMP Documents Overview	6
2. Terminology	7
3. Structure of the Configuration Data Model	9
3.1. Metering Process Decomposition in Selection Process and Cache	10
3.2. UML Representation	12
3.3. Exporter Configuration	16
3.4. Collector Configuration	18
4. Configuration Parameters	19
4.1. ObservationPoint Class	19
4.2. SelectionProcess Class	21
4.2.1. Selector Class	22
4.2.2. Sampler Classes	23
4.2.3. Filter Classes	24
4.3. Cache Class	26
4.3.1. ImmediateCache Class	27
4.3.2. TimeoutCache, NaturalCache, and PermanentCache Class	27
4.3.3. CacheLayout Class	29
4.4. ExportingProcess Class	32
4.4.1. SctpExporter Class	33
4.4.2. UdpExporter Class	35
4.4.3. TcpExporter Class	37
4.4.4. FileWriter Class	37
4.4.5. Options Class	38
4.5. CollectingProcess Class	40
4.5.1. SctpCollector Class	41
4.5.2. UdpCollector Class	42
4.5.3. TcpCollector Class	43
4.5.4. FileReader Class	44
4.6. Transport Layer Security Class	45
4.7. Transport Session Class	48
4.8. Template Class	52
5. Adaptation to Device Capabilities	53
6. YANG Module of the IPFIX/PSAMP Configuration Data Model	56
7. Examples	102
7.1. PSAMP Device	102
7.2. IPFIX Device	113

7.3. Export of Flow Records and Packet Reports	116
7.4. Collector and File Writer	119
7.5. Deviations	119
8. Security Considerations	120
9. IANA Considerations	122
Appendix A. Acknowledgements	122
10. References	123
10.1. Normative References	123
10.2. Informative References	123
Authors' Addresses	126

1. Introduction

Editor's note (to be removed prior to publication): This draft is to be published as RFC after ietf-ipfix-psamp-mib has become RFC. The RFC Editor is asked to replace references to ietf-ipfix-psamp-mib by references to the corresponding RFC. In the YANG module (Section 6), occurrences of "yyyy" shall be replaced by the RFC number of draft-ietf-ipfix-psamp-mib. In the YANG module (Section 6) and in Section 9, occurrences of "xxxx" shall be replaced by the RFC number of this document.

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, 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 storing and managing 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-20040204]. An XML document conforming to the configuration data model contains the configuration data of one Monitoring Device.

The configuration data model is designed for being used 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]. [RFC5815] 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 are 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]. [I-D.ietf-ipfix-psamp-mib] specifies the PSAMP MIB module as an

extension of the IPFIX SELECTOR MIB module defined in [RFC5815].

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:

- o 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
- o Definitions adopted from [RFC5103]:
 - * Reverse Information Element
- o Definitions adopted from [RFC5655]:
 - * File Reader
 - * File Writer
- o 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 multiple 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 which 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 [RFC5815] and the PSAMP MIB module [I-D.ietf-ipfix-psamp-mib] 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 which 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 performed by a Cache. Figure 1 illustrates this separation with the example of a basic Metering Process.

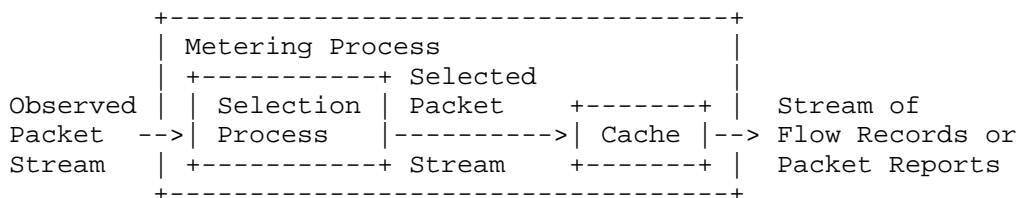


Figure 1: Selection Process and Cache forming a Metering Process

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 Primitive Selector) or an ordered composition of multiple Selectors (called 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 which selects all packets ("SelectAll" class in Section 4.2.1).

The configuration data model enables the configuration of a Selection Process which 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.

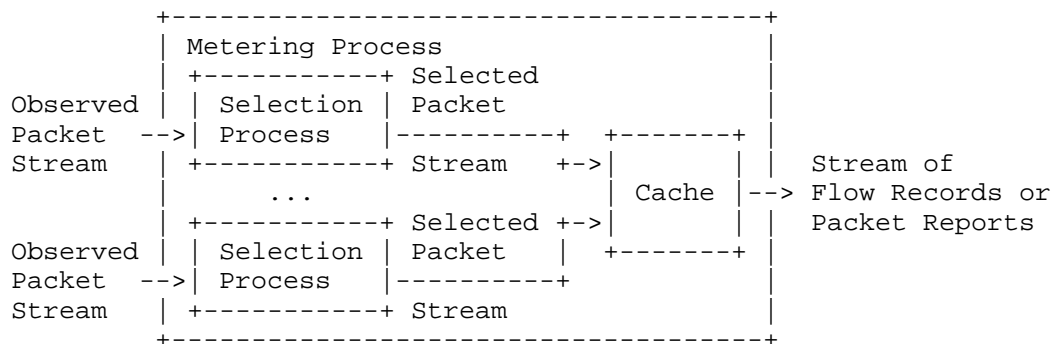


Figure 2: Metering Process with multiple Selection Processes

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.

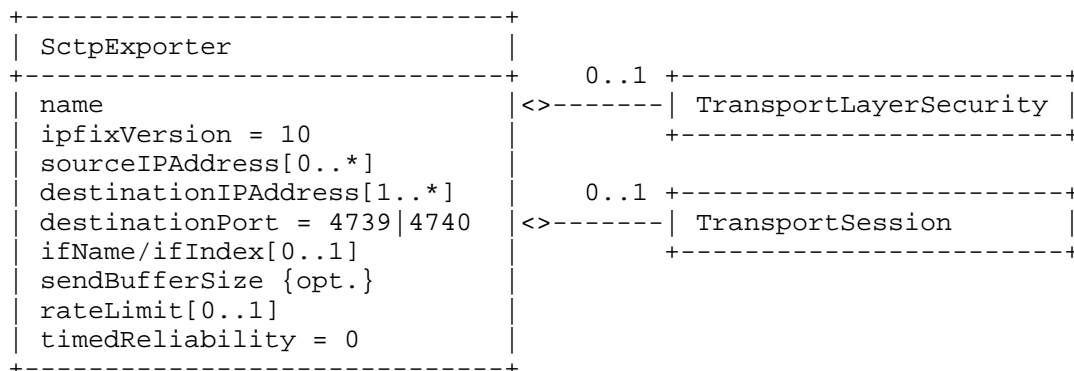


Figure 3: UML example: SctpExporter class

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 which 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 multi-homed 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 endowed with a default value which 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. 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}" specifying state parameters which 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 upper case character while the associated XML elements start with lower case characters. Parameters appear as XML elements which 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.

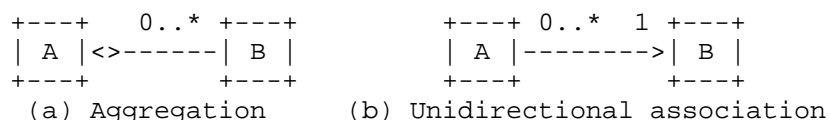


Figure 4: Class relationships in UML class diagrams

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 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 which contain configuration parameters and which 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 which 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 which 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 as 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 which are involved in the configuration of an IPFIX or PSAMP Exporter. The role of the classes can be briefly summarized as

follows:

- o The ObservationPoint class specifies an Observation Point (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.
- o The SelectionProcess class contains the configuration and state parameters of a Selection Process. The Selection Process may be composed of a single Selector or a sequence of Selectors, defining a Primitive or Composite Selector, respectively.

The Selection Process selects packets from one or more Observed Packet Streams, each originating from a different Observation Point. 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.

- o 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 multiple Exporting Process. Therefore, the Cache class enables references to one or multiple objects of the ExportingProcess class. If a Cache object does not specify any reference to an ExportingProcess object, the Cache output is dropped.

- o 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].

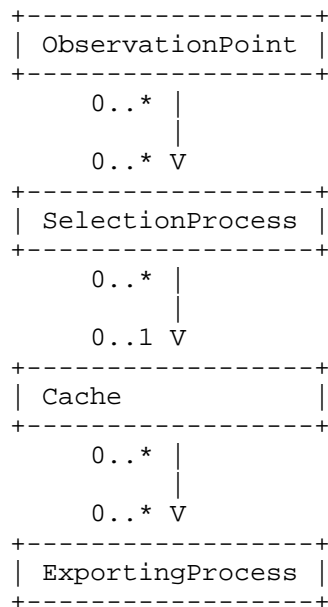


Figure 5: Class diagram of Exporter configuration

3.4. Collector Configuration

Figure 6 below shows the main classes of the configuration data model which 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 multiple 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.

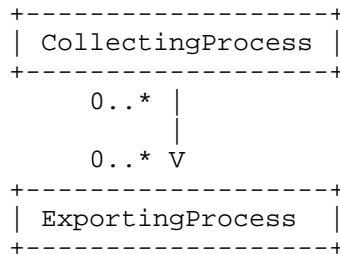


Figure 6: Class diagram of Collector configuration

4. Configuration Parameters

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

4.1. ObservationPoint Class

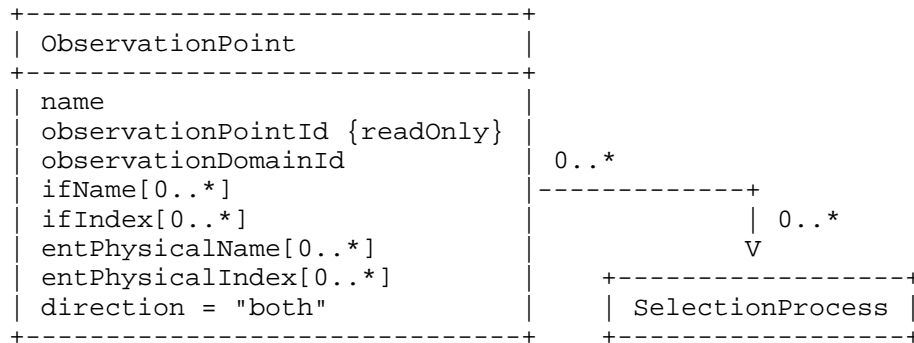


Figure 7: ObservationPoint class

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 modules [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` [RFC5102]) 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 [RFC5815].

`ifName/ifIndex/entPhysicalName/entPhysicalIndex`: These parameters identify interfaces and physical entities (e.g., linecards) on the Monitoring Device which 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 [RFC5815].

`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 multiple

SelectionProcess objects configuring Selection Processes that process the observed packets in parallel.

4.2. SelectionProcess Class

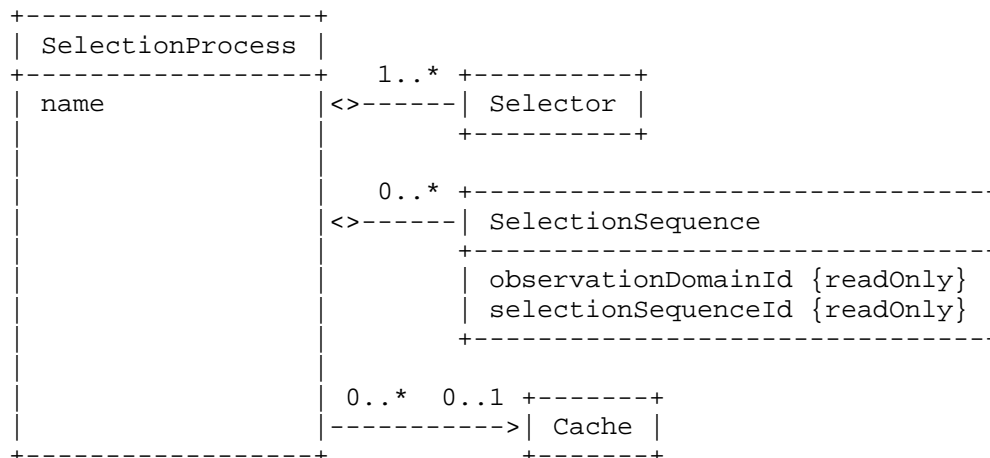


Figure 8: SelectionProcess class

Figure 8 shows the SelectionProcess class. The SelectionProcess class contains the configuration and state parameters of a Selection Process which 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 packet 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 which 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 the list of 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 specifications [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

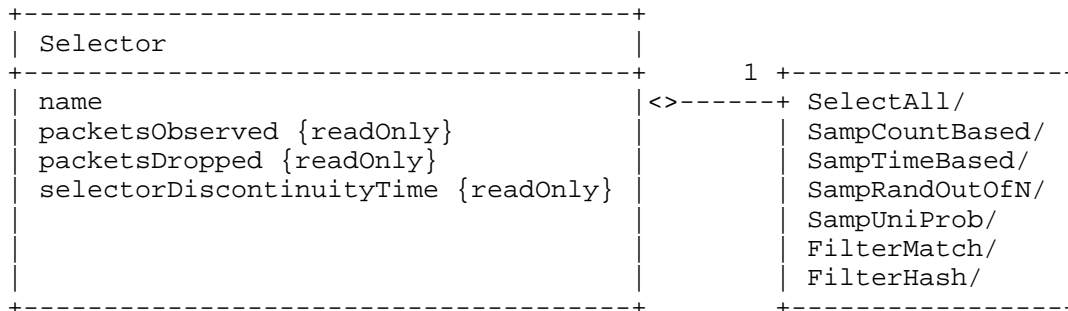


Figure 9: Selector class

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 [RFC5815]:

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

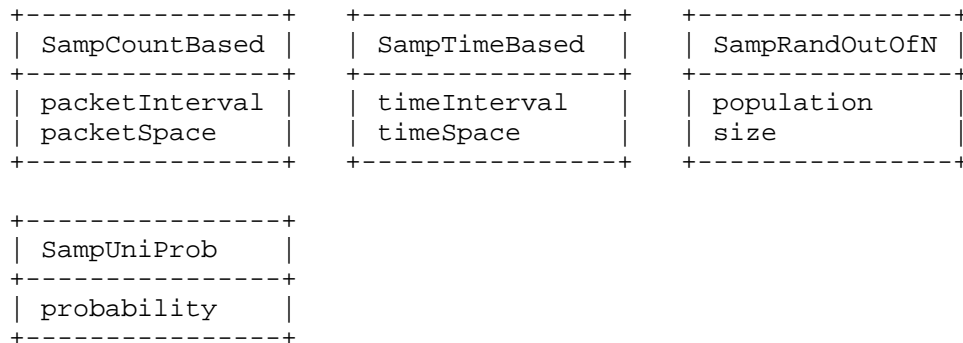


Figure 10: Sampler classes

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` [I-D.ietf-ipfix-psamp-mib].

`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` [I-D.ietf-ipfix-psamp-mib]. 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` [I-D.ietf-ipfix-psamp-mib].

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` [I-D.ietf-ipfix-psamp-mib].

4.2.3. Filter Classes

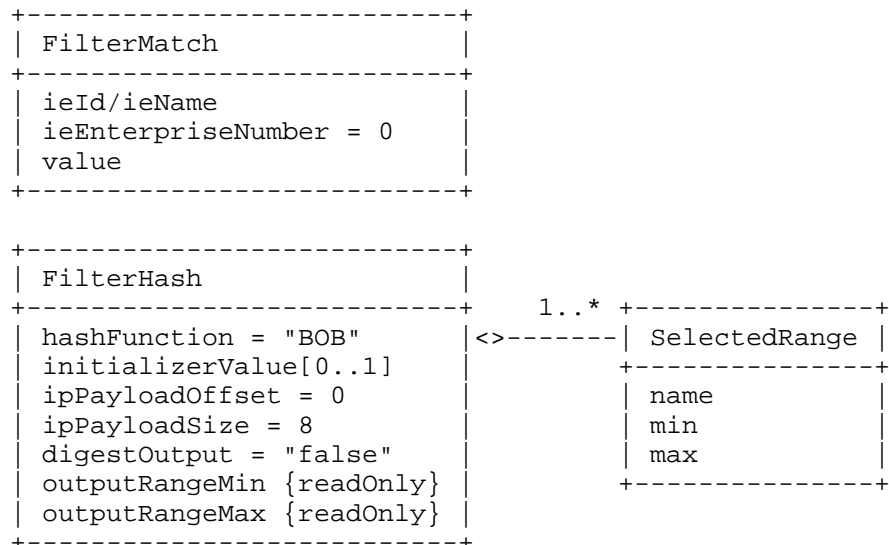


Figure 11: Filter classes

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 ID 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.

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 [I-D.ietf-ipfix-psamp-mib].

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 [I-D.ietf-ipfix-psamp-mib]. 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 6.2.5.6. These parameters correspond to the Information Elements hashIPPayloadOffset and hashIPPayloadSize [RFC5477], as well as to the PSAMP MIB objects psampFiltHashIpPayloadOffset and psampFiltHashIpPayloadSize [I-D.ietf-ipfix-psamp-mib].

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 [I-D.ietf-ipfix-psamp-mib].

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` [I-D.ietf-ipfix-psamp-mib].

4.3. Cache Class

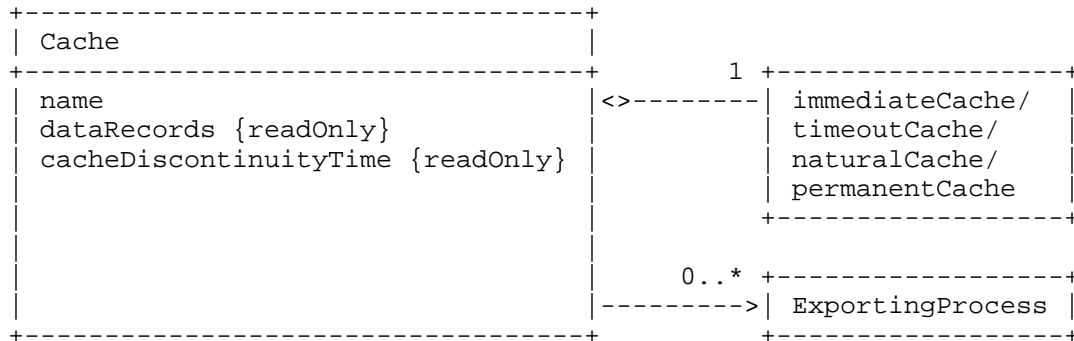


Figure 12: Cache class

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 Section 4.3.1 and Section 4.3.2. The following two state parameters are common to all Caches and therefore included in the Cache class itself:

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 [RFC5815].

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 [RFC5815].

A Cache object MAY refer to one or multiple `ExportingProcess` objects configuring different Exporting Processes.

4.3.1. ImmediateCache Class

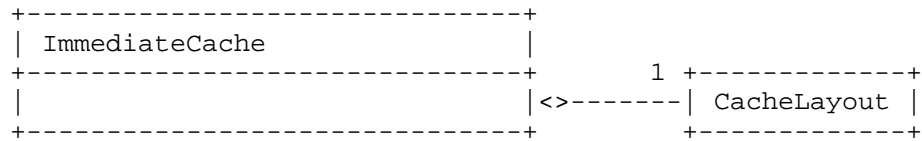


Figure 13: ImmediateCache class

The ImmediateCache class depicted in Figure 13 is used to configure a Cache which 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

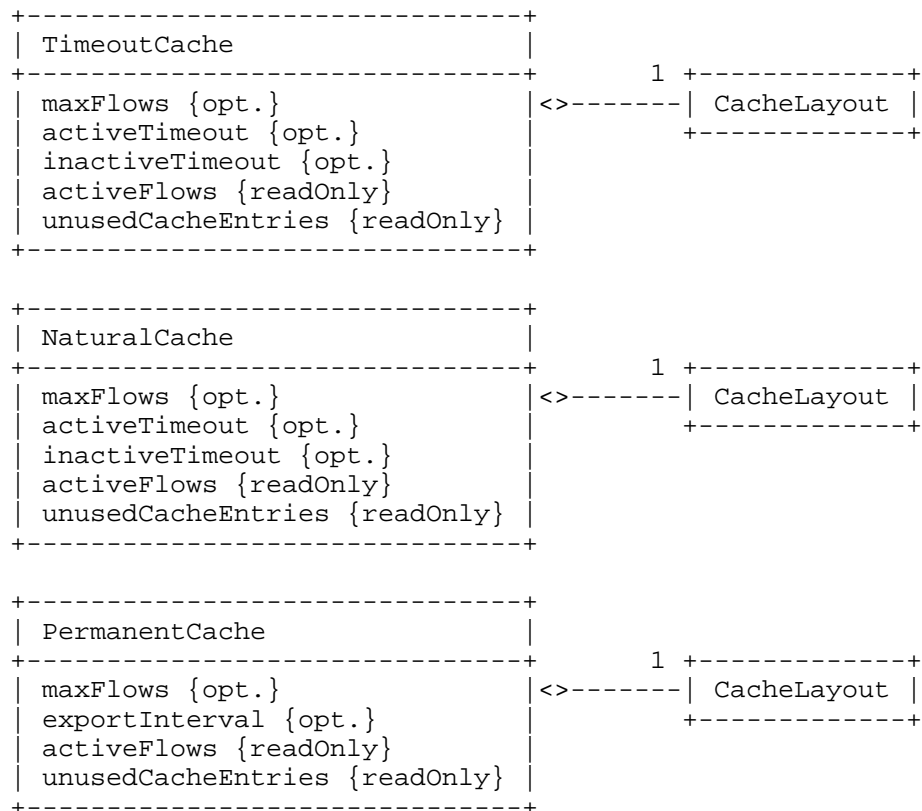


Figure 14: TimeoutCache, NaturalCache, and PermanentCache class

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

- o TimeoutCache: Flows expire after active or inactive timeout.
- o NaturalCache: Flows expire after active or inactive timeout, or on natural termination (e.g., TCP FIN, or TCP RST) of the Flow.
- o 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 which 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 [RFC5815].

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 [RFC5815].

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

[RFC5815].

inactiveTimeout: 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 inactive timeout. If not configured by the user, the Monitoring Device sets this parameter.
Note that this parameter corresponds to `ipfixMeteringProcessCacheInactiveTimeout` in the IPFIX MIB module [RFC5815].

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

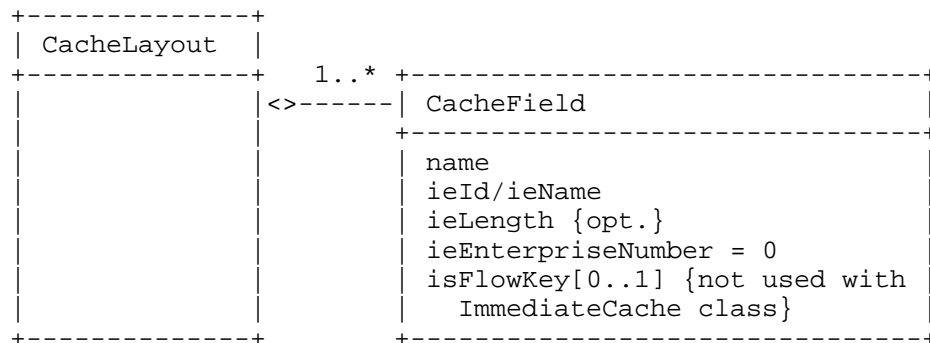


Figure 15: `CacheLayout` class

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 [RFC5102]), the field MUST be included in all Packet Reports which 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 [RFC5102]).

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 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. 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 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

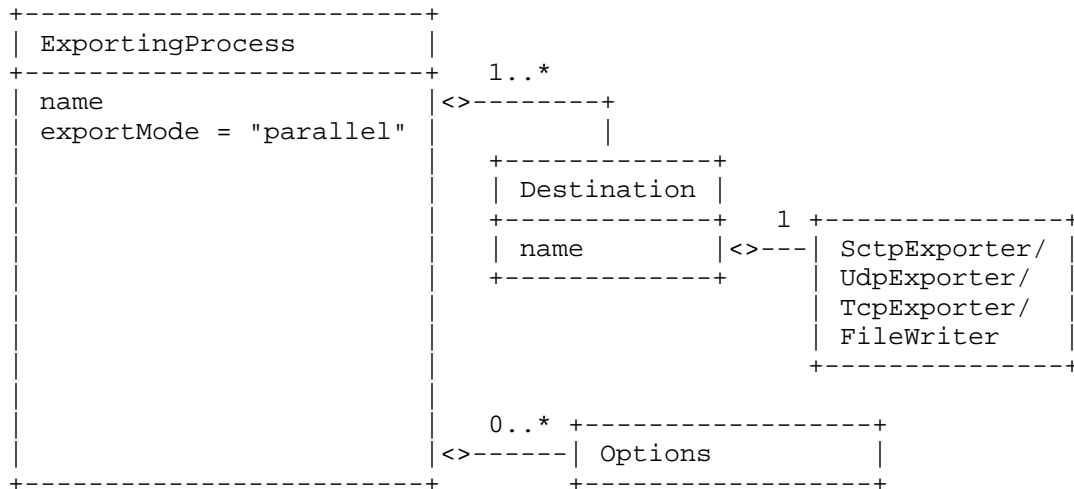


Figure 16: ExportingProcess class

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 Section 4.4.1, Section 4.4.2, Section 4.4.3, and Section 4.4.4.

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 [RFC5815]. 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 which refers to the primary destination is set to primary(1); the ipfixExportMemberType values which 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

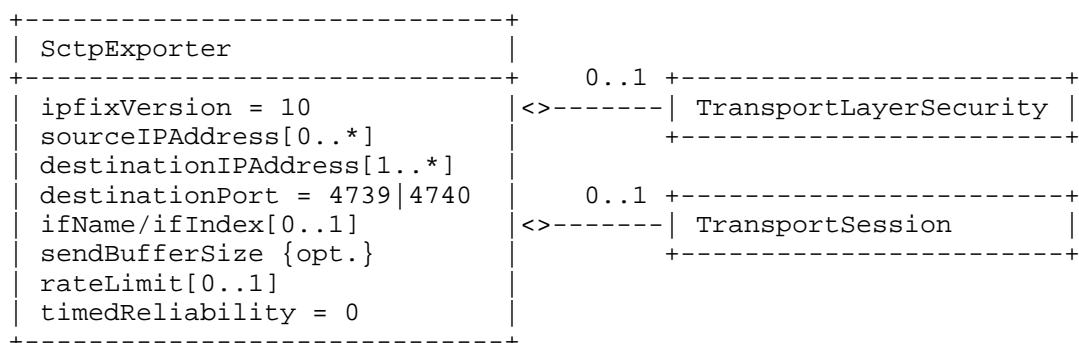


Figure 17: SctpExporter class

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 multi-homed SCTP endpoint. If not configured, all locally assigned IP addresses are eligible local IP addresses.

destinationIPAddress: One or multiple 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 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 [I-D.ietf-ipfix-export-per-sctp-stream].

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

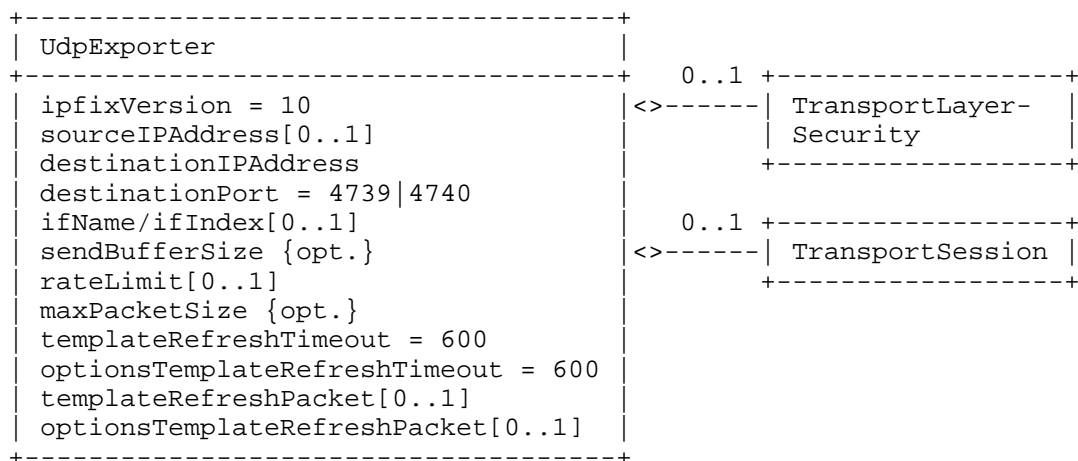


Figure 18: UdpExporter class

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 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 [RFC5815].

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 **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

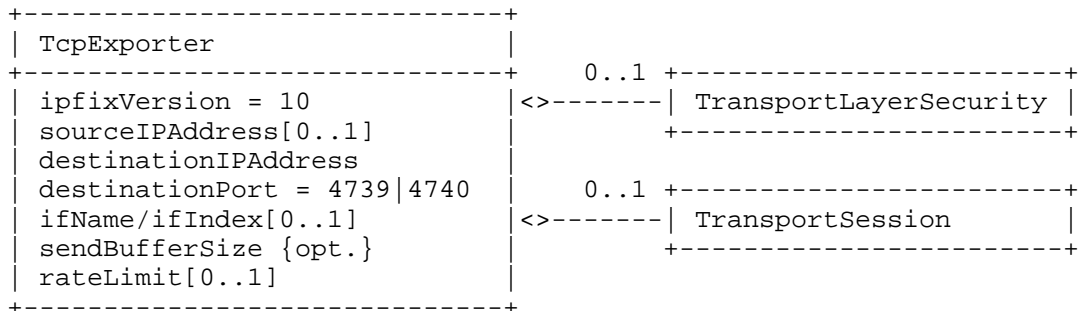


Figure 19: TcpExporter class

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

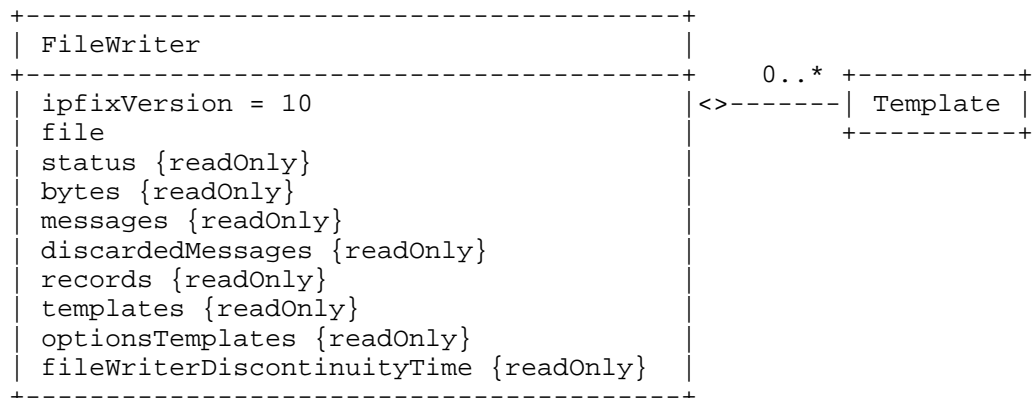


Figure 20: FileWriter classes

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

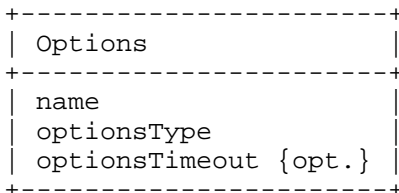


Figure 21: Options class

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 which 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

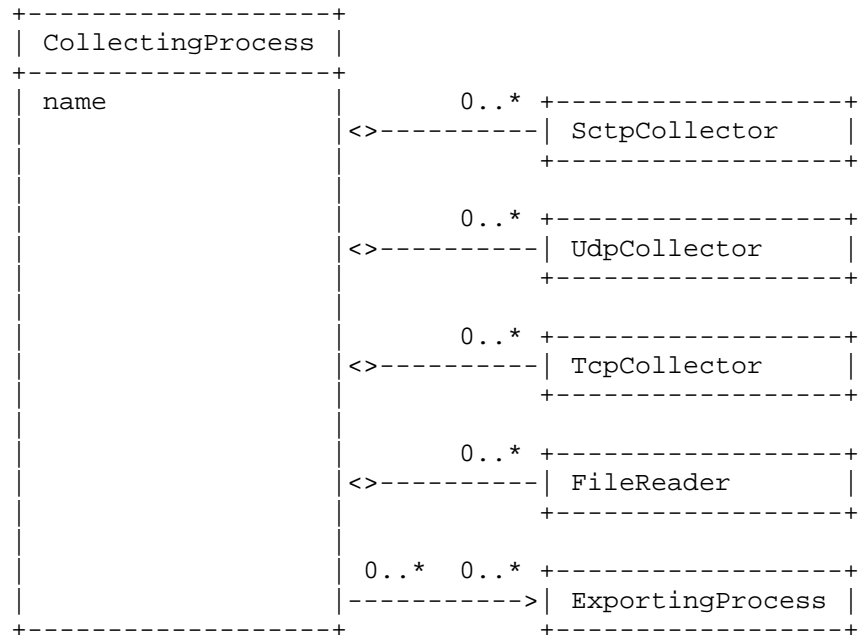


Figure 22: CollectingProcess class

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 Section 4.5.1, Section 4.5.2, Section 4.5.3, and Section 4.5.4.

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

4.5.1. SctpCollector Class

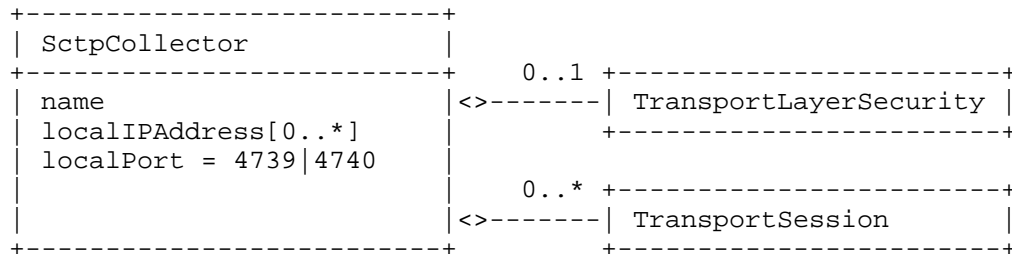


Figure 23: SctpCollector class

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 multi-homed 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, datagram transport layer security (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

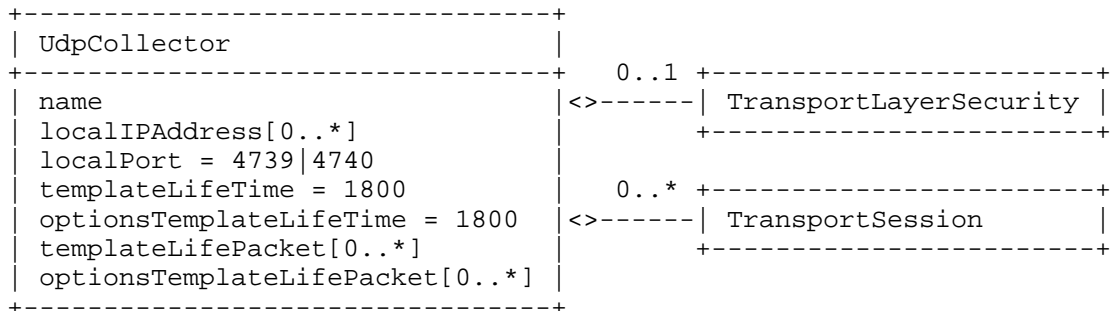


Figure 24: UdpCollector class

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 which 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 [RFC5815].

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 [RFC5815].

Using the TransportLayerSecurity class described in Section 4.6, datagram transport layer security (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

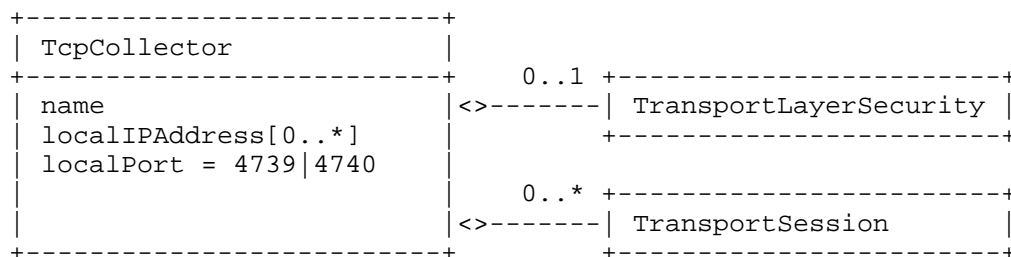


Figure 25: TcpCollector class

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, transport layer security (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

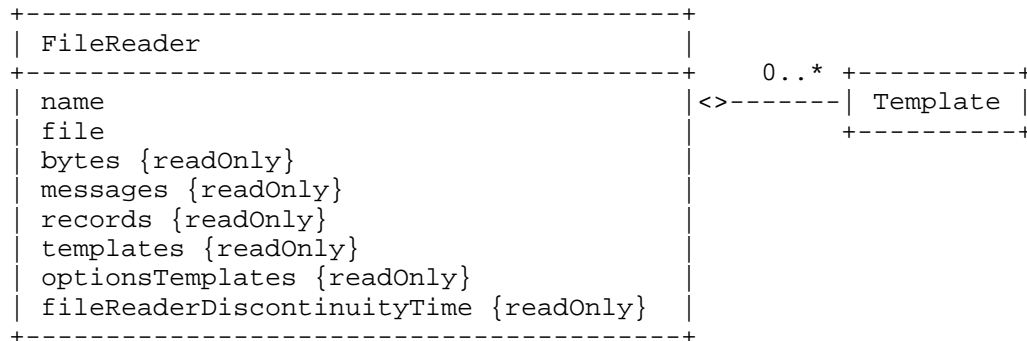


Figure 26: FileReader classes

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

```

+-----+
| TransportLayerSecurity |
+-----+
| localCertificationAuthorityDN[0..*] |
| localSubjectDN[0..*] |
| localSubjectFQDN[0..*] |
| remoteCertificationAuthorityDN[0..*] |
| remoteSubjectDN[0..*] |
| remoteSubjectFQDN[0..*] |
+-----+

```

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 transport layer security for IPFIX. Transport layer security can be enabled without configuring any additional parameters. In this case, an empty XML element `<transportLayerSecurity />` appears in the configuration. If transport layer security is enabled, the endpoint must use DTLS [RFC4347] 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:

"IPFIX Exporting Processes and IPFIX Collecting Processes are identified by the fully qualified domain name of the interface on which IPFIX Messages are sent or received, for purposes of X.509 client and server certificates as in [RFC5280].

To prevent man-in-the-middle attacks from impostor Exporting or Collecting Processes, the acceptance of data from an unauthorized Exporting Process, or the export of data to an unauthorized Collecting Process, strong mutual authentication via asymmetric keys MUST be used for both TLS and DTLS. Each of the IPFIX Exporting and Collecting Processes MUST verify the identity of its peer against its authorized certificates, and MUST verify that the peer's certificate matches its fully qualified domain name, or, in the case of SCTP, the fully qualified domain name of one of its endpoints.

The fully qualified domain name used to identify an IPFIX Collecting Process or Exporting Process may be stored either in a subjectAltName extension of type `dNSName`, or in the most specific

Common Name field of the Subject field of the X.509 certificate. If both are present, the subjectAltName extension is given preference."

In order to use transport layer security, 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 transport layer security is enabled on a Monitoring Device which 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 fully qualified domain names 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 multiple 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 authority. 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 multiple times to restrict the identification of the local endpoint during the TLS/DTLS handshake to certificates issued for specific subjects or for specific fully qualified domain names. Each occurrence of localSubjectDN contains a distinguished name identifying the local endpoint. Each occurrence of localSubjectFQDN contains a fully qualified domain name 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 fully qualified domain names 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 fully qualified domain name.

`remoteCertificationAuthorityDN`: This parameter MAY appear one or multiple 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 authority. 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 multiple times to restrict the authentication of remote endpoints during the TLS/DTLS handshake to certificates issued for specific subjects or for specific fully qualified domain names. Each occurrence of `remoteSubjectDN` contains a distinguished name identifying a remote endpoint. Each occurrence of `remoteSubjectFQDN` contains a fully qualified domain name which 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 fully qualified domain names 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 fully qualified domain name of the delivered certificate.

4.7. Transport Session Class

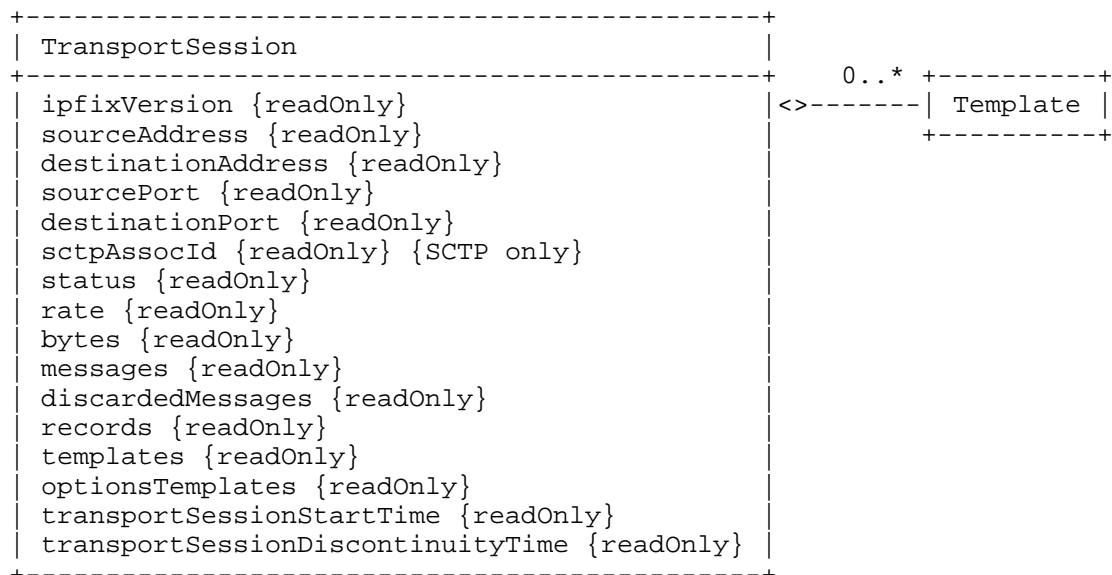


Figure 28: TransportSession class

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 [RFC5815]. An exception is the usage of the parameters `sourceAddress` and `destinationAddress`. If SCTP is transport protocol, Exporter or Collector **MAY** be multi-homed 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 which 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 [RFC5815].

sourceAddress, destinationAddress: If TCP or UDP is transport protocol, sourceAddress contains the IP address of the Exporter; destinationAddress contains the IP addresses of the Collector. Hence, the two parameters have identical values as ipfixTransportSessionSourceAddress and ipfixTransportSessionDestinationAddress in the IPFIX MIB module [RFC5815].
If SCTP is 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 which 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 [RFC5815].

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 [RFC5815].

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 [RFC5815].

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 [RFC5815].

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 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.

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

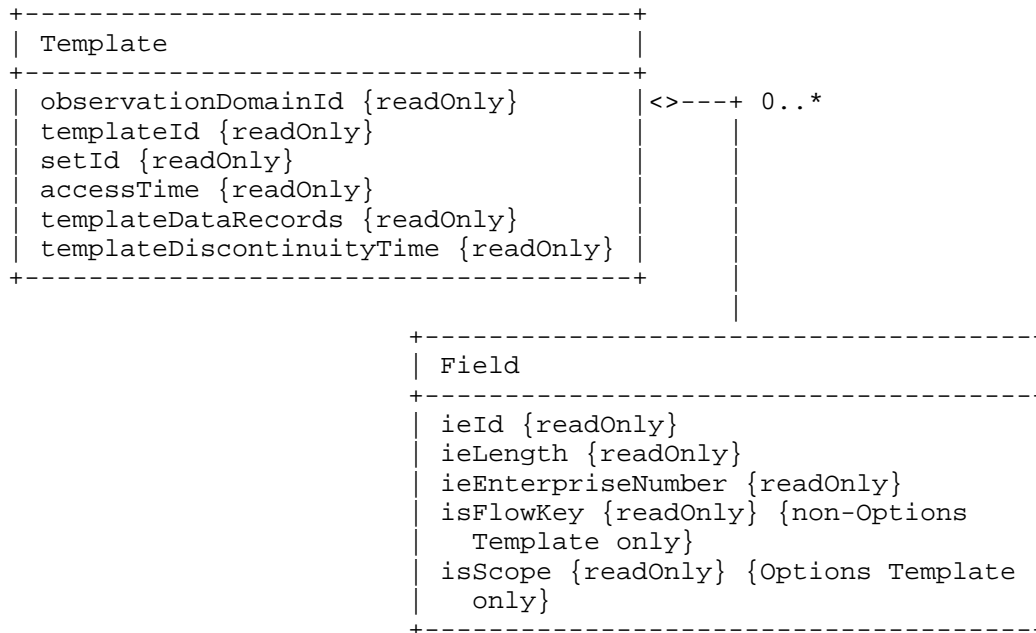


Figure 29: Template class

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 [RFC5815]:

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 ID, 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 [RFC5815].

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 which

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 which 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 which 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 which is not supported. The Monitoring Device MUST NOT silently accept configuration data which 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@2011-07-07.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:  <mailto:ipfix@ietf.org>
```

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

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

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

description

"IPFIX/PSAMP Configuration Data Model

Copyright (c) 2010 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 2011-07-07 {
 description "Version of
 draft-ietf-ipfix-configuration-model-10.";
 reference "RFCxxxx: IPFIX/PSAMP Configuration Data Model";
}

/*****
* 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 transport protocol.";
}

feature tcpTransport {
    description "If supported, the Monitoring Device supports TCP
    as 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 "RFC5475, Section 6.2.4.1.";
}
identity IPSX {
    base "hashFunction";
```

```
        description "IPSX hash function";
        reference "RFC5475, Section 6.2.4.1.";
    }
    identity CRC {
        base "hashFunction";
        description "CRC hash function";
        reference "RFC5475, 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 "RFC5476, Sections 6.5.1 and 6.5.2.";
}
identity selectionStatistics {
    base "optionsType";
    description "Selection Sequence Statistics Report.";
    reference "RFC5476, Sections 6.5.3.";
}
identity accuracy {
    base "optionsType";
    description "Accuracy Report.";
    reference "RFC5476, Section 6.5.4.";
}
identity reducingRedundancy {
    base "optionsType";
    description "Enables the utilization of Options Templates to
        reduce redundancy in the exported Data Records.";
    reference "RFC5473.";
}
identity extendedTypeInfo {
    base "optionsType";
    description "Export of extended type information for
        enterprise-specific Information Elements used in the
        exported Templates.";
    reference "RFC5610.";
}

/*****
* 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 "RFC5102, 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 "RFC2863 (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 "RFC5815, 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 "RFC5102, Section 5.1.10.";
    }
}
```

```
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 ID belong to the
        same Observation Domain.
        Note that this parameter corresponds to
        ipfixObservationPointObservationDomainId in the IPFIX MIB
        module.";
    reference "RFC5101; RFC5815, 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 1229; RFC5815, 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; RFC5815, 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 "RFC5475, Section 5.1; RFC5476, 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 "RFC5477, Section 8.2.2; RFCyyyy, 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 "RFC5477, Section 8.2.3; RFCyyyy, Section 6
            (psampSampCountBasedSpace).";
    }
}

grouping sampTimeBasedParameters {
    description "Configuration parameters of a Selector applying
        systematic time-based packet sampling to the packet
```

```
    stream.";
    reference "RFC5475, Section 5.1; RFC5476, 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 "RFC5477, Section 8.2.4; RFCyyyy, 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 "RFC5477, Section 8.2.5; RFCyyyy, Section 6
            (psampSampTimeBasedSpace).";
    }
}

grouping sampRandOutOfNParameters {
    description "Configuration parameters of a Selector applying
        n-out-of-N packet sampling to the packet stream.";
    reference "RFC5475, Section 5.2.1; RFC5476, 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 "RFC5477, Section 8.2.6; RFCyyyy, 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 "RFC5477, Section 8.2.7; RFCyyyy, 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 "RFC5475, Section 5.2.2.1;
        RFC5476, 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 "RFC5477, Section 8.2.8; RFCyyyy, 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 Information
        Element.";
    reference "RFC5475, Section 6.1; RFC5476, Section 6.5.2.5.";
    choice nameOrId {
        mandatory true;
        description "The field to be matched is specified by
            either the name or the ID of the Information
            Element.";
        leaf ieName {
            type ieNameType;
        }
    }
}
```

```
        description "Name of the Information Element.";
    }
    leaf ieId {
        type ieIdType;
        description "ID 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 "RFC5102.";
}
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 "RFC5475, Section 6.2; RFC5476, Section 6.5.2.6.";
    leaf hashFunction {
        type identityref {
            base "hashFunction";
        }
        default BOB;
        description "Hash function to be applied. According to
            RFC5475, 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 "RFCyyyy, 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 "RFC5477, Section 8.3.9; RFCyyyy, 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 RFC5476, Section
        6.2.5.6.
        This parameter corresponds to the Information Element
        hashIPPayloadOffset and to psampFiltHashIpPayloadOffset
        in the PSAMP MIB module.";
        reference "RFC5477, Section 8.3.2; RFCyyyy, 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 RFC5476,
        Section 6.2.5.6.
        This parameter corresponds to the Information Element
        hashIPPayloadSize and to psampFiltHashIpPayloadSize
        in the PSAMP MIB module.";
        reference "RFC5477, Section 8.3.3; RFCyyyy, 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 "RFC5477, 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 "RFC5477, Section 8.3.4; RFCyyyy, 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 "RFC5477, Section 8.3.5; RFCyyyy, 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 "RFC5477, Section 8.3.6; RFCyyyy, 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 "RFC5477, Section 8.3.7; RFCyyyy, 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 which 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 "RFC5815, 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 "RFC5815, 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 "RFC5815, 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 ID of the Information Element.";
      reference "RFC5102.";
      leaf ieName {
        type ieNameType;
        description "Name of the Information Element.";
      }
      leaf ieId {
        type ieIdType;
        description "ID 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 "RFC5101, Section 6.2; RFC5102.";
  }
  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
        RFC5103.";
    reference "RFC5101; RFC5102; RFC5103.";
}
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) or if the Cache Mode is 'immediate'.";
    }
    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, no additional
        Flows can be measured before any of the existing entries is
        removed. However, traffic which 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 "RFC5815, Section 8
(ipfixMeteringProcessCacheActiveTimeout).";
}
leaf inactiveTimeout {
  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 inactive timeout.
If not configured by the user, the Monitoring Device sets
this parameter.
Note that this parameter corresponds to
ipfixMeteringProcessCacheInactiveTimeout in the IPFIX
MIB module.";
  reference "RFC5815, Section 8
(ipfixMeteringProcessCacheInactiveTimeout).";
}
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 "RFC5815, 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 "RFC5815, Section 8
        (ipfixMeteringProcessCacheUnusedCacheEntries).";
}
}

grouping exportingProcessParameters {
    description "Parameters of an Exporting Process.";
    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 are 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 an export destination which 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 transport layer security and 4740 if transport
      layer security 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 1229.";
  leaf ifIndex {
    type uint32;
    description "Index of an interface as stored in the ifTable
      of IF-MIB.";
    reference "RFC 1229.";
  }
  leaf ifName {
    type string;
    description "Name of an interface as stored in the ifTable
      of IF-MIB.";
    reference "RFC 1229.";
  }
}
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 "RFC5476, 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 "Transport layer security 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 multi-homed 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 multiple 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 "RFC5101, Section 10.3.6; RFC5815, 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 "RFC5101, Section 10.3.6; RFC5815, 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 "RFC5101, Section 10.3.6; RFC5815, 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 "RFC5101, Section 10.3.6; RFC5815, 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) Template 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 which 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
      transport layer security and 4740 if transport layer
      security 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 "Transport layer security 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
      multi-homed 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 which are not received again within the configured
      lifetime become invalid at the Collecting Process.
      As specified in RFC5101, 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 "RFC5101, Section 10.3.7; RFC5815, 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 which are not received again within the
configured lifetime become invalid at the Collecting
Process.
As specified in RFC5101, 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 "RFC5101, Section 10.3.7; RFC5815, 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 "RFC5101, Section 10.3.7; RFC5815, 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 "RFC5101, Section 10.3.7; RFC5815, 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 "Transport layer security parameters.";
  leaf-list localCertificationAuthorityDN {
    type string;
    description "Distinguished names of certification authorities
      whose certificates may be used to identify the local
      endpoint.";
    reference "RFC5280.";
  }
}
```

```
leaf-list localSubjectDN {
  type string;
  description "Distinguished names which may be used in the
    certificates to identify the local endpoint.";
  reference "RFC5280.";
}
leaf-list localSubjectFQDN {
  type inet:domain-name;
  description "Fully qualified domain names which may be used to
    in the certificates to identify the local endpoint.";
  reference "RFC5280.";
}
leaf-list remoteCertificationAuthorityDN {
  type string;
  description "Distinguished names of certification authorities
    whose certificates are accepted to authorize remote
    endpoints.";
  reference "RFC5280.";
}
leaf-list remoteSubjectDN {
  type string;
  description "Distinguished names which are accepted in
    certificates to authorize remote endpoints.";
  reference "RFC5280.";
}
leaf-list remoteSubjectFQDN {
  type inet:domain-name;
  description "Fully qualified domain name which are accepted in
    certificates to authorize remote endpoints.";
  reference "RFC5280.";
}
}

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 "RFC5101; RFC5815, 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 "RFC5815, Section 8
      (ipfixTemplateObservationDomainId).";
  }
}
```

```
}
leaf templateId {
  type uint16 {
    range "256..65535" {
      description "Valid range of Template IDs.";
      reference "RFC5101";
    }
  }
  description "This number indicates the Template Id in the IPFIX
    message.
    Note that this parameter corresponds to ipfixTemplateId in
    the IPFIX MIB module.";
  reference "RFC5815, 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 "RFC5815, 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 "RFC5815, 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 "RFC5815, 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 "RFC5815, 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 Id of the field.
      Note that this parameter corresponds to
      ipfixTemplateDefinitionIeId in the IPFIX MIB module.";
    reference "RFC5815, Section 8 (ipfixTemplateDefinitionIeId);
      RFC5102.";
  }
  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 "RFC5815, Section 8
      (ipfixTemplateDefinitionIeLength); RFC5102.";
  }
  leaf ieEnterpriseNumber {
    type uint32;
    description "This parameter indicates the IANA enterprise
      number of the authority defining the Information Element
      Id.
      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 "RFC5815, Section 8
      (ipfixTemplateDefinitionIeEnterpriseNumber).";
  }
}
```

```
    }
    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 "RFC5815, 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 "RFC5815, Section 8
        (ipfixTemplateDefinitionFlags).";
    }
  }
}

grouping transportSessionParameters {
  description "State parameters of a Transport Session originating
    from an Exporting or terminating at a Collecting Process.
    Parameter names and semantics correspond to the managed
    objects in IPFIX-MIB.";
  reference "RFC5101; RFC5815, 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 "RFC5815, 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 which 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 "RFC5815, Section 8
        (ipfixTransportSessionSourceAddressType,
        ipfixTransportSessionSourceAddress);
        RFC4960, 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 which 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 "RFC5815, Section 8
        (ipfixTransportSessionDestinationAddressType,
        ipfixTransportSessionDestinationAddress);
        RFC4960, 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 "RFC5815, 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 "RFC5815, 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 "RFC5815, Section 8
        (ipfixTransportSessionSctpAssocId);
        RFC3871";
    }
    leaf status {
        type transportSessionStatus;
        description "Status of the Transport Session.
        Note that this parameter corresponds to
        ipfixTransportSessionStatus in the IPFIX MIB module.";
        reference "RFC5815, 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 "RFC5815, 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 "RFC5815, 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 "RFC5815, 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 "RFC5815, 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 "RFC5815, 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 "RFC5815, 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 "RFC5815, 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 "RFC5815, 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) Template 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
    which 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 "RFC5476.";
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 which 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 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 "RFC5815, 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 "RFC5815, 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 inactive
timeout; generation of Flow Records.";
    uses flowCacheParameters;
    uses cacheLayoutParameters;
  }
  container naturalCache {
    if-feature naturalCache;
    description "Flow expiration after active and inactive
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 Selectors 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 transport layer security parameters specify that the collector must supply a certificate for the fully qualified domain name 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>
  </selectionProcess>
</ipfix>
```

```
    </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 (ID = 313, length = 64)
observationTimeMilliseconds (ID = 322, length = 8)
```

Two Options Template 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 (ID = 301, length = 8)
observationPointId (ID = 138, length = 4)
selectorId (ID = 302, length = 4)
```

```
Template ID: 258
Scope: selectionSequenceId (ID = 301, length = 8)
observationPointId (ID = 138, length = 4)
selectorId (ID = 302, length = 4)
selectorId (ID = 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 (ID = 302, length = 4)
selectorAlgorithm (ID = 304, length = 2)
protocolIdentifier (ID = 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 (ID = 302, length = 4)
selectorAlgorithm (ID = 304, length = 2)
samplingSize (ID = 319, length = 4)
samplingPopulation (ID = 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 (ID = 301, length = 8)
selectorIdTotalPktsObserved (ID = 318, length = 8)
selectorIdTotalPktsSelected (ID = 319, length = 8)
```

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

After a short runtime, 100 packets have been observed at the two Observations 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>
  <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>
  <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>
  </observationPoint>
</ipfix>

```

```
<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>
    <inactiveTimeout>10000</inactiveTimeout>
  <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>transportProtocol</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>
</exportingProcess>
```

```

    <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 which 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>
    <inactiveTimeout>10</inactiveTimeout>
  <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 which 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, transport layer security 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 transport layer security 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 transport layer security 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: RFCxxxx

Appendix A. Acknowledgements

The authors thank Martin Bjorklund, Andy Bierman, and Ladislav Lhotka for helping specifying 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.

10. References

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

10.2. 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.

- [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.
- [RFC4347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security", RFC 4347, April 2006.
- [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.
- [RFC5815] Dietz, T., Kobayashi, A., Claise, B., and G. Muenz, "Definitions of Managed Objects for IP Flow Information Export", RFC 5815, April 2010.
- [RFC6110] Lhotka, L., "Mapping YANG to Document Schema Definition Languages and Validating NETCONF Content", RFC 6110, February 2011.
- [I-D.ietf-ipfix-psamp-mib]
Dietz, T., Claise, B., and J. Quittek, "Definitions of Managed Objects for Packet Sampling",
draft-ietf-ipfix-psamp-mib-03 (work in progress),
March 2011.
- [I-D.ietf-ipfix-export-per-sctp-stream]
Claise, B., Aitken, P., Johnson, A., and G. Muenz, "IPFIX Export per SCTP Stream",
draft-ietf-ipfix-export-per-sctp-stream-08 (work in progress), May 2010.
- [W3C.REC-xml-20040204]
Paoli, J., Maler, E., Yergeau, F., Sperberg-McQueen, C., and T. Bray, "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>>.
- [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>>.

[YANG-WEB]

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

[IANA-IPFIX]

"IANA Registry of IPFIX Information Elements",
Homepage <http://www.iana.org/assignments/ipfix/ipfix.xhtml>.

Authors' Addresses

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

Email: muenz@net.in.tum.de
URI: <http://www.net.in.tum.de/~muenz>

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

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: October 25, 2012

S. D'Antonio
University of Napoli
"Parthenope"
T. Zseby
CAIDA/FhG FOKUS
C. Henke
Tektronix Communication Berlin
L. Peluso
University of Napoli
April 23, 2012

Flow Selection Techniques
draft-ietf-ipfix-flow-selection-tech-11.txt

Abstract

Flow selection is the process of selecting a subset of flows from all observed flows. The Flow Selection Process may be located at an observation point, or on an IPFIX Mediator. Flow selection reduces the effort of post-processing flow data and transferring Flow Records. This document describes motivations for flow selection and presents flow selection techniques. It provides an information model for configuring flow selection techniques and discusses what information about a flow selection process should be exported.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of this Memo

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

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

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

This Internet-Draft will expire on October 25, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Scope	4
2. Terminology	4
3. Difference between Flow Selection and Packet Selection	7
4. Flow selection as a Function in the IPFIX Architecture	8
4.1. Flow selection during the Metering Process	10
4.2. Flow selection during the Exporting Process	10
4.3. Flow selection as a function of the IPFIX Mediator	10
5. Flow Selection Techniques	11
5.1. Flow Filtering	11
5.1.1. Property Match Filtering	11
5.1.2. Hash-based Flow Filtering	12
5.2. Flow Sampling	12
5.2.1. Systematic sampling	12
5.2.2. Random Sampling	13
5.3. Flow-state Dependent Flow Selection	13
5.4. Flow-state Dependent Packet Selection	14
6. Configuration of Flow Selection Techniques	14
6.1. Flow Selection Parameters	16
6.2. Description of Flow-state Dependent Packet Selection	18
7. Information Model for Flow Selection Configuration and Reporting	18
7.1. flowSelectorAlgorithm	20
7.2. flowSelectedOctetDeltaCount	21
7.3. flowSelectedPacketDeltaCount	21
7.4. flowSelectedFlowDeltaCount	21
7.5. selectorIDTotalFlowsObserved	22
7.6. selectorIDTotalFlowsSelected	22
7.7. samplingFlowInterval	22
7.8. samplingFlowSpace	23
7.9. flowSamplingTimeInterval	23
7.10. flowSamplingTimeSpace	24
7.11. hashFlowDomain	24
8. IANA Considerations	24
8.1. Registration of Information Elements	24
8.2. Registration of Object Identifier	32
9. Security Considerations	32
10. Acknowledgments	34
11. References	34
11.1. Normative References	34
11.2. Informative References	34
Authors' Addresses	35

1. Scope

This document describes flow selection techniques for network traffic measurements. A flow is defined as a set of packets with common properties as described in [RFC5101]. Flow selection can be done to limit the resource demands for capturing, storing, exporting and post-processing of Flow Records. It also can be used to select a particular set of flows that are of interest to a specific application. This document provides a categorization of flow selection techniques and describes configuration and reporting parameters for them. In order to be compliant with this document, at least one of the flow selection schemes MUST be implemented. That means that the configuration parameters as well as the reporting Information Elements for this particular scheme MUST be supported.

This document also addresses configuration and reporting parameters for flow-state dependent packet selection as described in [RFC5475], although this technique is categorized as packet selection. The reason is that flow-state dependent packet selection techniques often aim at the reduction of resources for flow capturing and flow processing. Furthermore, they were only briefly discussed in [RFC5475]. Therefore we included configuration and reporting considerations for such techniques in this document.

2. Terminology

This document is consistent with the terminology introduced in [RFC5101], [RFC5470], [RFC5475] and [RFC3917]. As in [RFC5101] and [RFC5476], the first letter of each IPFIX-specific and PSAMP-specific term is capitalized along with the flow selection specific terms defined here.

* Packet Classification

Packet Classification is a process by which packets are mapped to specific Flow Records based on packet properties or external properties (e.g. interface). The properties (e.g. header information, packet content, AS number) make up the Flow Key. In case a Flow Record for a specific Flow Key already exists the Flow Record is updated, otherwise a new Flow Record is created.

* Packet Aggregation Process

In the IPFIX Metering Process the Packet Aggregation Process aggregates packet data into flow data and forms the Flow Records. After the aggregation step only the aggregated flow information is available. Information about individual packets is lost.

* Flow Selection Process

A Flow Selection Process takes Flow Records as its input and selects a subset of this set as its output. A Flow Selection Process MAY run in several places within the IPFIX architecture. A Flow Selection Process MAY be part of an IPFIX Metering Process, Exporting Process or as an Intermediate Selection Process as defined for the IPFIX Mediator [RFC6183].

* Flow Selection State

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

- (i) sequence number of packets and accounted Flow Records;
- (ii) number of selected flows;
- (iii) number of observed flows;
- (iv) current flow cache occupancy;
- (v) flow specific counters, lower and upper bounds;
- (vi) flow selection timeout intervals.

* Flow Selector

A Flow Selector defines the action of a Flow Selection Process on a single flow of its input. The Flow Selector can make use of the following information in order to establish whether a flow has to be selected or not:

- (i) the content of the Flow Record;
- (ii) any state information related to the Metering Process or Exporting Process;
- (iii) any Flow Selection State that may be maintained by the Flow Selection Process.

* Complete Flow

A Complete Flow consists of all the packets that enter the Flow Selection Process within the flow time-out interval, and which belong to the same flow as defined by the flow definition in

[RFC5470]. For this definition only packets that arrive at the Flow Selection Process are considered. That means, packets that are not observed at the Flow Selection Process because of prior packet selection or packet loss are not considered as belonging to the Complete Flow.

* Flow Filtering

Flow Filtering selects flows based on a deterministic function on the Flow Record content, Flow Selection State, external properties (e.g. ingress interface) or external events (e.g. violated Access Control List). If the relevant parts of the Flow Record content can already be observed at packet level (e.g. Flow Keys from packet header fields) Flow Filtering can be performed at packet level by Property Match Filtering as described in [RFC5475].

* Hash-based Flow Filtering

Hash-based Flow Filtering is a deterministic flow filter function that selects flows based on a Hash Function. The Hash Function is calculated over parts of the Flow Record content or external properties which are called the Hash Domain. If the hash value falls into a predefined Hash Selection Range the flow is selected. Hash-based Flow Filtering can already be applied at packet level, in which case the Hash Domain MUST contain the Flow Key of the packet. In case Hash-based Flow Filtering is used to select the same subset of flows at different observation points, the Hash Domain MUST comprise parts of the packet or flow that are invariant on the packet/flow path. Also refer to the according Trajectory Sampling Application Example on packet level in [RFC5475]

* Flow-state Dependent Flow Selection

Flow-state Dependent Flow Selection is a selection function that selects or drops flows based on the current Flow Selection State. The selection can be either deterministic, random or non-uniform random.

* Flow-state Dependent Packet Selection

Flow-state Dependent Packet Selection is a selection function that selects or drops packets based on the current Flow Selection State. The selection can be either deterministic, random or non-uniform random. Flow-state Dependent Packet Selection can be used to prefer the selection of packets belonging to specific flows. For example the selection probability of packets belonging to flows that are already within the Flow Cache may be higher than

for packets that have not been recorded yet.

* Flow Sampling

Flow Sampling selects flows based on Flow Record sequence or arrival times (e.g. entry in flow cache, arrival time at Exporter or Mediator). The selection can be systematic (e.g. every n-th flow) or based on a random function (e.g. select each Flow Record with probability p, or randomly select n out of N Flow Records).

3. Difference between Flow Selection and Packet Selection

Flow selection differs from packet selection described in [RFC5475]. Packet selection techniques consider packets as the basic element and the parent population consists of all packets observed at an observation point. In contrast to this the basic elements in flow selection are the flows. The parent population consists of all observed flows and the selection process operates on the flows. The major characteristics of flow selection are the following:

- Flow selection takes flows as basic elements. For packet selection, packets are considered as basic elements.
- Flow selection can only take place after Packet Classification, because the classification rules determine to which flow a packet belongs. Packet selection can be applied before and after Packet Classification.
- Flow selection operates on Complete Flows. That means that after the Flow Selection Process either all packets of the flow are kept or all packets of the flow are discarded. That means that if the flow selection is preceded by a packet selection process the Complete Flow consists only of the packets that were not discarded during the packet selection.

There are some techniques that are difficult to unambiguously categorize into one of the categories. Here we give some guidance how to categorize such techniques:

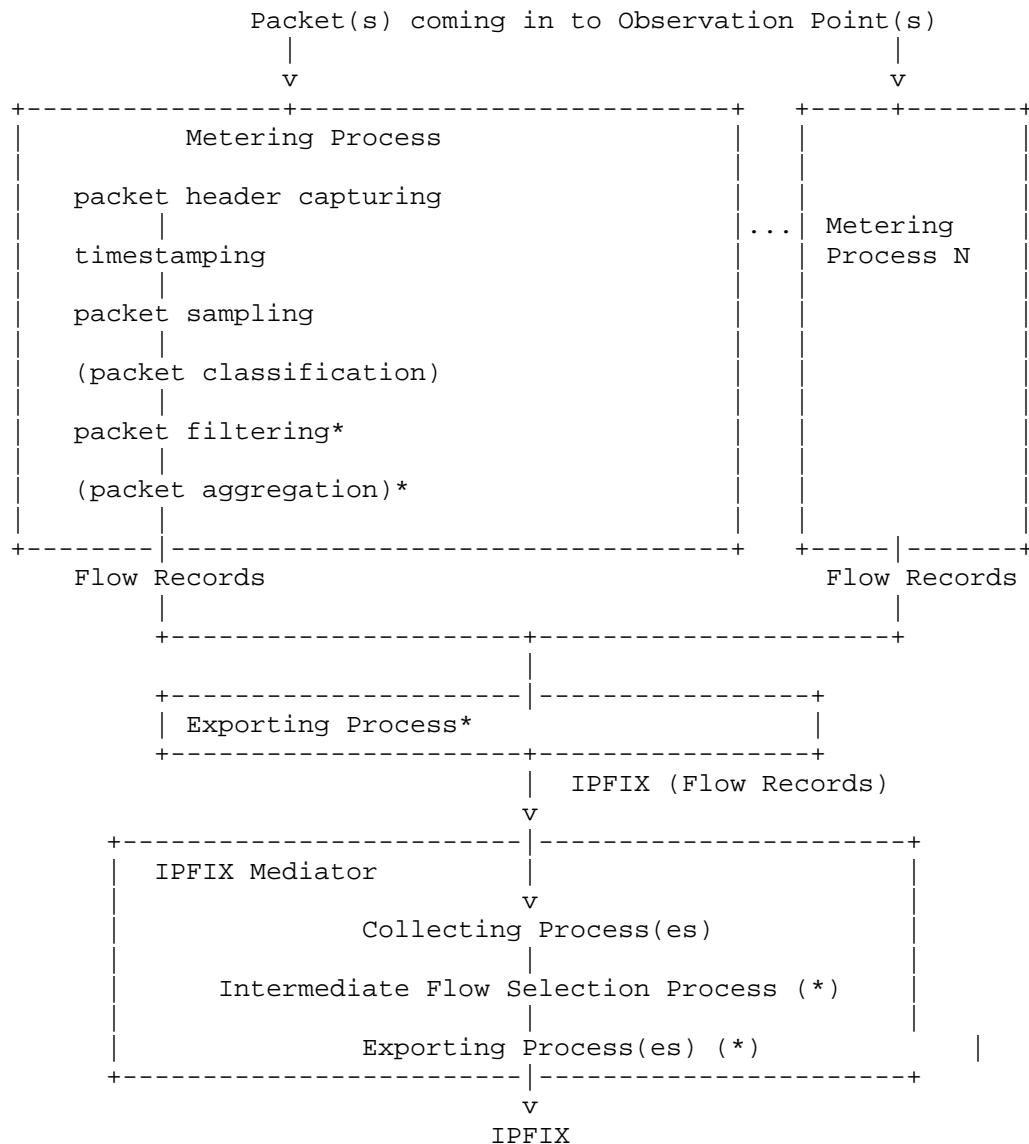
- Techniques that can be considered as both packet and flow selection: some packet selection techniques result in the selection of Complete Flows and therefore can be considered as packet or as flow selection at the same time. An example is Property Match Filtering of all packets to a specific destination address. If flows are defined based on destination addresses, such a packet selection also results in a flow selection and can be considered as packet or flow

selection.

- Flow-state Dependent Packet Selection (as described in [RFC5475]): there exist techniques that select packets based on the flow state, e.g. based on the number of already observed packets belonging to the flow. Examples of these techniques from the literature are "Sample and Hold" [EsVa01] "Fast Filtered Sampling" [MSZC10] or the "Sticky Sampling" algorithm presented in [MaMo02]. Such techniques can be used to influence which flows are captured (e.g. increase the selection of packets belonging to large flows) and reduce the number of flows that need to be stored in the flow cache. Nevertheless, such techniques do not necessarily select Complete Flows, because they do not ensure that all packets of a selected flow are captured. Therefore Flow-state Dependent Packet Selection methods that do not ensure that either all or no packets of a flow are selected strictly speaking have to be considered as packet selection techniques and not as flow selection techniques.

4. Flow selection as a Function in the IPFIX Architecture

Figure 1 shows the IPFIX reference model as defined in [RFC5470] and shows the Packet Classification and Packet Aggregation Process in the Metering Process.



(*) indicates where flow selection can take place.

Figure 1: Flow selection in the IPFIX Architecture

In contrast to packet selection, flow selection is always applied after the packets are classified into flows. Flows can be selected at different stages of the measurement chain:

1. during the Metering Process
2. during Exporting Process(es)
3. during an Intermediate Selection Process on a Mediator

4.1. Flow selection during the Metering Process

In the Packet Aggregation Process the packet information is used to update the Flow Records in the flow cache. Flow selection that is applied before aggregation equals a packet selection process. The flow still consists of individual packets. Those are then selected based on the classification information, i.e. based on the flow they belong to. Flow selection before aggregation can be based on the fields of the Flow Key (also on a hash value over these fields), but not based on characteristics that are only available after packet aggregation (e.g. flow size, flow duration). Flow selection during the Metering Process is applied to reduce resources for all succeeding processes or to select specific flows of interest in case such flow characteristics are already observable at packet level (e.g. flows to specific IP addresses). In contrast, Flow-state Dependent Packet Selection is a packet selection method, because it does not necessarily select Complete Flows.

4.2. Flow selection during the Exporting Process

The Flow Selection Process at the Exporter is similar to an Intermediate Selection Process as described in [RFC6183] and works on Flow records. Flow selection during the Exporting Process can therefore also depend on flow characteristics that are only visible after the aggregation of packets, such as flow size and flow duration. The Exporting Process may implement policies for exporting only a subset of the Flow Records which have been stored in the system memory in order to unload flow export and flow post-processing. Flow selection during the Exporting Process may select only the subset of Flow Records which are of interest to the users application, or select only as many Flow Records as can be handled by the available resources (e.g. limited export link capacity).

4.3. Flow selection as a function of the IPFIX Mediator

As shown in Figure 1, flow selection can be performed as an Intermediate Process within an IPFIX Mediator [RFC6183]. The Intermediate Selection Process takes Flow Record stream as its input and selects Flow Records from a sequence based upon criteria-evaluated record values. The Intermediate Selection Process can again apply a flow selection technique to obtain flows of interest to the application. Further, the Intermediate Selection Process can

base its selection decision on the correlation of data from different observation points, e.g. by only selecting flows that were at least recorded on two observation points.

5. Flow Selection Techniques

A flow selection technique selects either all or none of the packets of a flow, otherwise the technique has to be considered as packet selection. We distinguish between Flow Filtering and Flow Sampling.

5.1. Flow Filtering

Flow Filtering is a deterministic function on the IPFIX Flow Record content. If the relevant flow characteristics are already observable at packet level (e.g. Flow Keys), Flow Filtering can be applied before aggregation at packet level. In order to be compliant with this document, at least the Property Match Filtering MUST be implemented.

5.1.1. Property Match Filtering

Property Match Filtering can be performed similarly to Property Match Filtering for packet selection described in [RFC5475]. The difference is that, instead of packet fields, Flow Record fields are here used to derive the selection decision. Property Match Filtering is typically used to select a specific subset of the flows that are of interest to a particular application (e.g. all flows to a specific destination, all large flows, etc.). Properties on which the filtering is based can be Flow Keys, Flow Timestamps, or Per-Flow Counters described in [RFC5102]. Examples of properties are the flow size in bytes, the number of packets in the flow, the observation time of the first or last packet, or the maximum packet length. An example is to select flows with more than a threshold number of observed octets. The selection criteria can be a specific value, a set of specific values, or an interval. For example, a flow is selected if destinationIPv4Address and the total number of packets of the flow equal two predefined values. Property Match Filtering can be applied during the Metering Process if the properties are already observable at the packet level (e.g. Flow Key fields). For example, a flow is selected if sourceIPv4Address and sourceIPv4PrefixLength equal, respectively, two specific values.

There are content-based Property Match Filtering techniques that require a computation on the current flow cache. An example is the selection of the largest flows or a percentage of flows with the longest lifetime. This type of Property Match Filtering is also used in flow selection techniques that react to external events (e.g.

resource constraint). For example when the flow cache is full, the Flow Record with the lowest flow volume per current flow life time may be deleted.

5.1.2. Hash-based Flow Filtering

Hash-based Flow Filtering uses a Hash Function h to map the Flow Key c onto a Hash Range R . A flow is selected if the hash value $h(c)$ is within the Hash Selection Range S , which is a subset of R . Hash-based Flow Filtering can be used to emulate a random sampling process but still enable the correlation between selected flow subsets at different observation points. Hash-based Flow Filtering is similar to Hash-based Packet Selection, and in fact is identical when Hash-based Packet Selection uses the Flow Key that defines the flow as the hash input. Nevertheless there may be the incentive to apply Hash-based Flow Filtering not on the packet level during the Metering Process, for example when the size of the selection range and therefore the sampling probability is dependent on the number of observed flows.

5.2. Flow Sampling

Flow Sampling operates on Flow Record sequence or arrival times. It can use either a systematic or a random function for the selection process. Flow Sampling usually aims at the selection of a representative subset of all flows in order to estimate characteristics of the whole set (e.g. mean flow size in the network).

5.2.1. Systematic sampling

Systematic sampling is a deterministic selection function. Systematic sampling may be a periodic selection of the N -th Flow Record which arrives at the Exporting or Intermediate Selection Process. Systematic sampling MAY be applied during the Metering Process. An example would be to create, besides the Flow cache of selected flows, an additional data structure that saves the Flow Keys of the flows that are not selected. The selection of a flow would then be based on the first packet of a flow. Everytime a packet belonging to a new flow (which is neither in the data structure of the selected or not selected flows) arrives at the measurement point, a counter is increased. In case the counter is increased to a multiple of N a new flow cache entry is created, and in case the counter is not a multiple of N the Flow Key is added to the data structure for not selected flows.

Systematic sampling can also be time-based. Time-based systematic sampling is applied by only creating flows that are observed between

time-based start and stop triggers. The time interval may be applied at packet level during the Metering Process or after aggregation on flow level, e.g. by selecting a flow arriving at the Exporting Process every n seconds.

5.2.2. Random Sampling

Random flow sampling is based on a random process which requires the calculation of random numbers. One can differentiate between n -out- N and probabilistic flow sampling.

5.2.2.1. n -out-of- N Flow Sampling

In n -out-of- N Sampling, n elements are selected out of the parent population that consists of N elements. One example would be to generate n different random numbers in the range $[1, N]$ and select all flows that have a flow position equal to one of the random numbers.

5.2.2.2. Probabilistic Flow Sampling

In probabilistic Sampling, the decision whether or not a flow is selected is made in accordance with a predefined selection probability. For probabilistic Sampling, the Sample Size can vary for different trials. The selection probability does not necessarily have to be the same for each flow. Therefore, we distinguish between uniform probabilistic sampling (with the same selection probability for all flows) and non-uniform probabilistic sampling (where the selection probability can vary for different flows). For non-uniform probabilistic Flow Sampling the sampling probability may be adjusted according to the Flow Record content. An example would be to increase the selection probability of large volume flows over small volume flows as described in the Smart Sampling technique [DuLT01].

5.3. Flow-state Dependent Flow Selection

Flow-state Dependent Flow Selection can be a deterministic or random flow selection process based on the Flow Record content and the flow state which may be kept additionally for each of the flows. External processes may update counters, bounds and timers for each of the Flow Records and the Flow Selection Process utilises this information for the selection decision. A review of Flow-state Dependent Flow Selection techniques that aim at the selection of the most frequent items by keeping additional flow state information can be found in [CoHa08]. Flow-state Dependent Flow Selection can only be applied after packet aggregation, when a packet has been assigned to a flow. The selection process then decides based upon the flow state for each flow if it is kept in the flow cache or not. Two Flow State Dependent Flow Selection Algorithms are here described:

The frequent algorithm [KaPS03] is a technique that aims at the selection of all flows that at least exceed a $1/k$ fraction of the Observed Packet Stream. The algorithm has only a flow cache of size $k-1$ and each flow in the cache has an additional counter. The counter is incremented each time a packet belonging to the flow in the flow cache is observed. In case the observed packet does not belong to any flow all counters are decremented and if any of the flow counters has a value of zero the flow is replaced with a flow formed from the new packet.

Lossy counting is a selection technique that identifies all flows whose packet count exceeds a certain percentage of the whole observed packet stream (e.g. 5% of all packets) with a certain estimation error ϵ . Lossy counting separates the observed packet stream in windows of size $N=1/\epsilon$, where N is an amount of consecutive packets. For each observed flow an additional counter will be held in the flow state. The counter is incremented each time a packet belonging to the flow is observed and all counters are decremented at the end of each window and all flows with a counter of zero are removed from the flow cache.

5.4. Flow-state Dependent Packet Selection

Flow-state Dependent Packet Selection is not a flow selection technique but a packet selection technique. Nevertheless we will describe configuration and reporting parameters for this technique in this document. An example is the "Sample and Hold" algorithm [EsVa01] that tries to prefer large volume flows in the selection. When a packet arrives it is selected when a Flow Record for this packet already exists. In case there is no Flow Record, the packet is selected by a certain probability that is dependent on the packet size.

6. Configuration of Flow Selection Techniques

This section describes the configuration parameters of the flow selection techniques presented above. It provides the basis for an information model to be adopted in order to configure the Flow Selection Process within an IPFIX Device. The actual information model with the Information Elements (IEs) for the configuration is described together with the reporting IEs in section 7. The following table gives an overview of the defined selection techniques, where they can be applied and what their input parameters are. Depending on where the flow selection techniques are applied different input parameters can be configured.

Overview of Flow Selection Techniques:

Location	Selection Method	Selection Input
During the Metering Process based on Packets	Flow-state Dependent Packet Selection	packet sampling probabilities, Flow Selection State, packet properties
	Property Match Flow Filtering	Flow record IEs, Selection Interval
	Hash-based Flow Filtering	selection range, Hash Function, Flow Key, (seed)
	Time-based Systematic Flow Sampling	flow position (derived from arrival time of packets), flow selection state
	Sequence-based Systematic Flow Sampling	flow position (derived from packet position), flow selection state
	Random Flow Sampling	random number generator or list and packet position, flow state
Exporting / Intermediate Selection Process	Property Match Flow Filtering	Flow Record content, filter function
	Hash-based Flow Filtering	selection range, Hash Function, hash input (Flow Keys and other flow properties)
	Flow-state Dependent Flow Selection	flow state parameters, random number generator or list
	Time-based Systematic Flow Sampling	flow arrival time, flow state
	Sequence-based Systematic Flow Sampling	flow position, flow state

	Random Flow Sampling	random number generator or list and flow position, flow state
--	-------------------------	---

Table 1: Overview of Flow Selection Techniques

6.1. Flow Selection Parameters

In this section, we define what parameters are required to describe the most common Flow Selection techniques.

Flow Selection Parameters:

For Property Match Filtering:

- Information Element as specified in [iana-ipfix-assignments]): Specifies the Information Element which is used as the property in the filter expression.
- Selection Value or Value Interval: Specifies the value or interval of the filter expression. Packets and Flow Record that have a value equal to the Selection Value or within the Interval will be selected.

For Hash-based Flow Filtering:

- Hash Domain: Specifies the bits from the packet or flow which are taken as the hash input to the Hash Function.
- Hash Function: Specifies the name of the Hash Function that is used to calculate the hash value. Possible Hash Functions are BOB [RFC5475], IPSX [RFC5475], CRC-32 [Bra75]
- Hash Selection Range: Flows that have a hash value within the Hash Selection Range are selected. The Hash Selection Range can be a value interval or arbitrary hash values within the Hash Range of the Hash Function.
- Random Seed or Initializer Value: Some Hash Functions require an initializing value. In order to make the selection decision more secure one can choose a random seed that configures the hash function.

For Flow-state Dependent Flow Selection:

- frequency threshold:
Specifies the frequency threshold s for flow state dependent flow selection techniques that try to find the most frequent items within a dataset. All flows which exceed the defined threshold will be selected.
- accuracy parameter:
specifies the accuracy parameter e for techniques that deal with the frequent items problems. The accuracy parameter defines the maximum error, i.e. no flows that have a true frequency less than $(s - e)N$ are selected, where s is the frequency threshold and N is the total number of packets.

The above list of parameters for Flow-state Dependent Flow Selection techniques is suitable for the presented frequent item and lossy counting algorithms. Nevertheless a variety of techniques exist with very specific parameters which are not defined here.

For Systematic time-based Flow Sampling:

- Interval length (in usec)
Defines the length of the sampling interval during which flows are selected.
- Spacing (in usec)
The spacing parameter defines the spacing in usec between the end of one sampling interval and the start of the next succeeding interval.

For Systematic count-based Flow Sampling:

- Interval length
Defines the number of flows that are selected within the sampling interval.
- Spacing
The spacing parameter defines the spacing in number of observed flows between the end of one sampling interval and the start of the next succeeding interval.

For random n-out-of-N Flow Sampling:

- Population Size N
The Population Size N is the number of all flows in the Population from which the sample is drawn.

- Sampling Size n
The sampling size n is the number of flows that are randomly drawn from the population N .

For probabilistic Flow Sampling:

- Sampling probability p
The sampling probability p defines the probability by which each of the observed flows is selected.

6.2. Description of Flow-state Dependent Packet Selection

The configuration of Flow-state Dependent Packet Selection has not been described in [RFC5475] therefore the parameters are defined here:

For Flow-state Dependent Packet Selection:

- packet selection probability per possible flow state interval
Defines multiple {flow interval, packet selection probability} value pairs that configure the sampling probability depending on the current flow state.
- additional parameters
For the configuration of flow state dependent packet selection additional parameters or packet properties may be required, e.g. the packet size ([EsVa01])

7. Information Model for Flow Selection Configuration and Reporting

In this section we describe Information Elements (IEs) that MUST be exported by a flow selection process in order to support the interpretation of measurement results from flow measurements where only some flows are selected. The information is mainly used to report how many packets and flows have been observed in total and how many of them were selected. This helps for instance to calculate the Attained Selection Fraction (see also [RFC5476]), which is an important parameter to provide an accuracy statement. The IEs can provide reporting information about Flow Records, packets or bytes. The reported metrics are total number of elements and the number of selected elements. From this the number of dropped elements can be derived. All counters SHOULD be exported and reset when a new measurement interval starts.

List of Flow Selection Information Elements:

ID	Name	ID	Name
301	selectionSequenceID	302	selectorID
TBD1	flowSelectorAlgorithm	1	octetDeltaCount
TBD2	flowSelectedOctetDeltaCount	2	packetDeltaCount
TBD3	flowSelectedPacketDeltaCount	3	originalFlowsPresent
TBD4	flowSelectedFlowDeltaCount	TBD5	selectorIDTotalFlowsObserved
TBD6	selectorIDTotalFlowsSelected	TBD7	samplingFlowInterval
TBD8	samplingFlowSpace	309	samplingSize
310	samplingPopulation	311	samplingProbability
TBD9	flowSamplingTimeInterval	TBD10	flowSamplingTimeSpace
326	digestHashValue	TBD11	hashFlowOffset
TBD12	hashFlowSize	329	hashOutputRangeMin
330	hashOutputRangeMax	331	hashSelectedRangeMin
332	hashSelectedRangeMax	333	hashDigestOutput
334	hashInitialiserValue	320	absoluteError
321	relativeError	336	upperCILimit
337	lowerCILimit	338	confidenceLevel

Table 2: Flow Selection Information Elements

7.1. flowSelectorAlgorithm

Description:

This Information Element identifies the flow selection method(e.g., Filtering, Sampling) that is applied by the Flow Selection Process. Most of these methods have parameters as described in Section 6. Further Information Elements are needed to fully specify packet selection with these methods and all their parameters. Further method identifiers may be added to the list below. It might be necessary to define new Information Elements to specify their parameters. The flowSelectorAlgorithm registry is maintained by IANA. New assignments for the registry will be administered by IANA and are subject to Expert Review [RFC5226]. The registry can be updated when specifications of the new method(s) and any new Information Elements are provided.

ID	Method	Parameters
1	Systematic count-based Sampling	flowSamplingInterval flowSamplingSpace
2	Systematic time-based Sampling	flowSamplingTimeInterval flowSamplingTimeSpace
3	Random n-out-of-N Sampling	samplingSize samplingPopulation
4	Uniform probabilistic Sampling	samplingProbability
5	Property Match Filtering	Information Element Value Range
Hash-based Filtering		hashInitialiserValue hashFlowDomain
6	using BOB	hashSelectedRangeMin hashSelectedRangeMax
7	using IPSX	hashOutputRangeMin hashOutputRangeMax
8	using CRC	
9	Flow State Dependent Flow Selection	No agreed Parameters

Abstract Data Type: unsigned16

ElementId: TBD1

Data Type Semantics: identifier

Status: Proposed

7.2. flowSelectedOctetDeltaCount

Description:

This Information Element specifies the volume in octets of all flows that are selected during the Flow Selection Process since the previous report.

Abstract Data Type: unsigned64

ElementId: TBD2

Units: Octets

Status: Proposed

7.3. flowSelectedPacketDeltaCount

Description:

This Information Element specifies the volume in packets of all flows that were selected during the Flow Selection Process since the previous report.

Abstract Data Type: unsigned64

ElementId: TBD3

Units: Packets

Status: Proposed

7.4. flowSelectedFlowDeltaCount

Description:

This Information Element specifies the number of Flows that were selected during the Flow Selection Process since the last report.

Abstract Data Type: unsigned64

ElementId: TBD4

Units: Flows

Status: Proposed

7.5. selectorIDTotalFlowsObserved

Description:

This Information Element specifies the total number of flows observed by a Selector, for a specific value of SelectorId. This Information Element should be used in an Options Template scoped to the observation to which it refers. See Section 3.4.2.1 of the IPFIX protocol document [RFC5101] .

Abstract Data Type: unsigned64

ElementId: TBD5

Units: Flows

Status: Proposed

7.6. selectorIDTotalFlowsSelected

Description:

This Information Element specifies the total number of flows selected by a Selector, for a specific value of SelectorId. This Information Element should be used in an Options Template scoped to the observation to which it refers. See Section 3.4.2.1 of the IPFIX protocol document [RFC5101].

Abstract Data Type: unsigned64

ElementId: TBD6

Units: Flows

Status: Proposed

7.7. samplingFlowInterval

Description:

This Information Element specifies the number of flows that are consecutively sampled. A value of 100 means that 100 consecutive

flows are sampled. For example, this Information Element may be used to describe the configuration of a systematic count-based Sampling Selector.

Abstract Data Type: unsigned64

ElementId: TBD7

Units: Flows

Status: Proposed

7.8. samplingFlowSpace

Description:

This Information Element specifies the number of flows between two "samplingFlowInterval"s. A value of 100 means that the next interval starts 100 flows (which are not sampled) after the current "samplingFlowInterval" is over. For example, this Information Element may be used to describe the configuration of a systematic count-based Sampling Selector.

Abstract Data Type: unsigned64

ElementId: TBD8

Units: Flows

Status: Proposed

7.9. flowSamplingTimeInterval

Description:

This Information Element specifies the time interval in microseconds during which all arriving flows are sampled. For example, this Information Element may be used to describe the configuration of a systematic time-based Sampling Selector.

Abstract Data Type: unsigned64

ElementId: TBD9

Units: microseconds

Status: Proposed

7.10. flowSamplingTimeSpace

Description:

This Information Element specifies the time interval in microseconds between two "flowSamplingTimeInterval"s. A value of 100 means that the next interval starts 100 microseconds (during which no flows are sampled) after the current "flowSamplingTimeInterval" is over. For example, this Information Element may be used to describe the configuration of a systematic time-based Sampling Selector.

Abstract Data Type: unsigned64

ElementId: TBD10

Units: microseconds

Status: Proposed

7.11. hashFlowDomain

Description:

This Information Element specifies the Information Elements that are used by the Hash-based flow Selection Selector as the Hash Domain.

Abstract Data Type: unsigned16

ElementId: TBD11

Data Type Semantics: identifier

Status: Proposed

8. IANA Considerations

8.1. Registration of Information Elements

IANA will register the following IEs in the IPFIX Information Elements registry at <http://www.iana.org/assignments/ipfix/ipfix.xml>:

Value	Name	Data Type	Data Type Semantics	Status	Description
1	flowSelectorAlgorithm	unsigned16	identifier	Proposed	This Information Element identifies the flow selection method(e.g., Filtering, Sampling) that is applied by the Flow Selection Process
2	flowSelectedOctetDeltaCount	unsigned64	Octets	Proposed	This Information Element specifies the volume in octets of all flows that are selected during the Flow Selection Process since the previous report.
3	flowSelectedPacketDeltaCount	unsigned64	Packets	Proposed	This Information Element specifies the volume in packets of all flows that were selected during the Flow Selection Process since the previous report.

4	flowSelectedFlowDeltaCount	unsigned64	Flows	Proposed	This Information Element specifies the number of Flows that were selected during the Flow Selection Process since the last report.
5	selectorIDTotalFlowsObserved	unsigned64	Flows	Proposed	This Information Element specifies the total number of flows observed by a Selector, for a specific value of SelectorId. This Information Element should be used in an Options Template scoped to the observation to which it refers. See Section 3.4.2.1 of the IPFIX protocol document [RFC5101]

6	selectorIDTotalFlowsSelected	unsigned64	Flows	Proposed	This Information Element specifies the total number of flows selected by a Selector, for a specific value of SelectorId. This Information Element should be used in an Options Template scoped to the observation to which it refers. See Section 3.4.2.1 of the IPFIX protocol document [RFC5101].
---	------------------------------	------------	-------	----------	---

7	samplingFlowInterval	unsigned64	Flows	Proposed	This Information Element specifies the number of flows that are consecutively sampled. A value of 100 means that 100 consecutive flows are sampled. For example, this Information Element may be used to describe the configuration of a systematic count-based Sampling Selector.
---	----------------------	------------	-------	----------	--

8	samplingFlowSpace	unsigned64	Flows	Proposed	This Information Element specifies the number of flows between two "samplingFlowInterval"s. A value of 100 means that the next interval starts 100 flows (which are not sampled) after the current "samplingFlowInterval" is over. For example, this Information Element may be used to describe the configuration of a systematic count-based Sampling Selector.
---	-------------------	------------	-------	----------	---

9	flowSamplingTimeInterval	unsigned64	microseconds	Proposed	This Information Element specifies the time interval in microseconds during which all arriving flows are sampled. For example, this Information Element may be used to describe the configuration of a systematic time-based Sampling Selector.
---	--------------------------	------------	--------------	----------	---

10	flowSamplingTimeSpace	unsigned64	microseconds	Proposed	This Information Element specifies the time interval in microseconds between two "flowSamplingTimeInterval"s. A value of 100 means that the next interval starts 100 microseconds (during which no flows are sampled) after the current "flowSamplingTimeInterval" is over. For example, this Information Element may be used to describe the configuration of a systematic time-based Sampling Selector.
11	hashFlowDomain	unsigned16	identifier	Proposed	This Information Element specifies the Information Elements that are used by the Hash-based flow Selection Selector as the Hash Domain.

Table 3: Flow Selection methods

8.2. Registration of Object Identifier

IANA will register the following OID in the IPFIX-SELECTOR-MIB Functions sub-registry at <http://www.iana.org/assignments/smi-numbers> according to the procedures set forth in [I-D.dkcm-ipfix-rfc5815bis]

Decimal	Name	Description	Reference
1	flowSelectorAlgorithm	This Object Identifier identifies the flow selection method (e.g., Filtering, Sampling) that is applied by the Flow Selection Process	[RFCyyyy]

Table 4: Information Elements to be registered

Editor's Note (to be removed prior to publication): the RFC editor is asked to replace "yyyy" in this document by the number of the RFC when the assignment has been made.

9. Security Considerations

The described flow sampling techniques and the hash-based flow filtering technique aim at the selection of a representative subset of flows in order to make an accurate estimation of the population. An adversary may have incentives to influence the selection of flows, for example to circumvent accounting.

Security considerations concerning the choice of a Hash Function for Hash-based Packet Selection have been discussed in Section 6.2.3 of [RFC5475] and are also appropriate for Hash-Based Flow Selection. This section discussed a number of potential attacks to craft Streams that are disproportionately detected and/or discover the Hash Function parameters, the vulnerabilities of different Hash Functions to these attacks, and practices to minimize these vulnerabilities.

For other sampling approaches a user can gain knowledge about the start and stop triggers in time-based systematic Sampling, e.g., by sending test packets. This knowledge might allow users to modify their send schedule in a way that their packets are disproportionately selected or not selected. For random Sampling, a cryptographically strong random number generator should be used in

order to prevent that an adversary can predict the selection decision [GoRe07].

Further security threats can occur when Sampling parameters are configured or communicated to other entities. The protocol(s) for the configuration and reporting of Sampling parameters are out of scope of this document. Nevertheless, a set of initial requirements for future configuration and reporting protocols are stated below:

1. Protection against disclosure of configuration information: Flow selection configuration information describes the selection process and its parameters. This information can be useful to attackers. Attackers may craft packets that never fit the selection criteria in order to prevent flows to be seen by the selection process. They can also craft a lot of packets that fit the selection criteria and overload or bias subsequent processes. Therefore any transmission of configuration data (e.g., to configure a process or to report its actual status) should be protected by encryption.
2. Protection against modification of configuration information: If wrong configuration information is sent to the flow selection process, it can lead to a malfunction of the selection process. Also if wrong configuration information is reported from the selection process to other processes it can lead to wrong estimations at subsequent processes. Therefore any protocol that transmits configuration information should prevent that an attacker can modify configuration information. Data integrity can be achieved by authenticating the data.
3. Protection against malicious nodes sending configuration information: The remote configuration of flow selection methods should be protected against access by unauthorized nodes. This can be achieved by access control lists at the selection devices and source authentication. The reporting of configuration data from a selection process has to be protected in the same way. That means that also protocols that report configuration data from the selection process to other processes need to protect against unauthorized nodes reporting configuration information.

The security threats that originate from communicating configuration information to and from selection processes cannot be assessed solely with the information given in this document. A further more detailed assessment of security threats is necessary when a specific protocol for the configuration or reporting configuration data is proposed.

10. Acknowledgments

We would like to thank the IPFIX group, especially Brian Trammell, Paul Aitken and Benoit Claise for fruitful discussions and for proofreading the 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.
- [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.

11.2. Informative References

- [Bra75] Brayer, K., "Evaluation of 32 Degree Polynomials in Error Detection on the SATIN IV Autovon Error Patterns", National Technical Information Service p.74, August 1975.
- [CoHa08] Cormode, G. and M. Hadjieleftheriou, "Finding frequent items in data streams", Journal, Proceedings of the Very Large DataBase Endowment VLDB Endowment, Volume 1 Issue 2, August 2008, August 2008.
- [DuLT01] Duffield, N., Lund, C., and M. Thorup, "Charging from Sampled Network Usage", ACM Internet Measurement Workshop IMW 2001, San Francisco, USA, November 2001.
- [EsVa01] Estan, C. and G. Varghese, "New Directions in Traffic Measurement and Accounting: Focusing on the Elephants, Ignoring the Mice", ACM SIGCOMM Internet Measurement Workshop 2001, San Francisco (CA), November 2001.

- [KaPS03] Karp, R., Papadimitriou, C., and S. S. Shenker, "A simple algorithm for finding frequent elements in sets and bags.", ACM Transactions on Database Systems, Volume 28, 51-55, 2003, March 2003.
- [MSZC10] Mai, J., Sridharan, A., Zang, H., and C. Chuah, "Fast Filtered Sampling", Computer Networks Volume 54, Issue 11, Pages 1885-1898, ISSN 1389-1286, January 2010.
- [MaMo02] Manku, G. and R. Motwani, "Approximate Frequency Counts over Data Streams", Proceedings of the International Conference on Very large DataBases (VLDB) pages 346--357, 2002, Hong Kong, China, 2002.
- [RFC3917] Quittek, J., Zseby, T., Claise, B., and S. Zander, "Requirements for IP Flow Information Export (IPFIX)", RFC 3917, October 2004.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.
- [RFC5470] Sadasivan, G., Brownlee, N., Claise, B., and J. Quittek, "Architecture for IP Flow Information Export", RFC 5470, March 2009.
- [RFC6183] Kobayashi, A., Claise, B., Muenz, G., and K. Ishibashi, "IP Flow Information Export (IPFIX) Mediation: Framework", RFC 6183, April 2011.
- [iana-ipfix-assignments] "IP Flow Information Export Information Elements", 2007, <<http://www.iana.org/assignments/ipfix/ipfix.xml>>.

Authors' Addresses

Salvatore D'Antonio
University of Napoli "Parthenope"
Centro Direzionale di Napoli Is. C4
Naples 80143
Italy

Phone: +39 081 5476766
Email: salvatore.dantonio@uniparthenope.it

Tanja Zseby
CAIDA/FhG FOKUS
San Diego Supercomputer Center (SDSC)
University of California, San Diego (UCSD)
9500 Gilman Drive
La Jolla CA 92093-0505
USA

Email: tanja@caida.org

Christian Henke
Tektronix Communication Berlin
Wohlrabedamm 32
Berlin 13629
Germany

Phone: +49 17 2323 8717
Email: christian.henke@tektronix.com

Lorenzo Peluso
University of Napoli
Via Claudio 21
Napoli 80125
Italy

Phone: +39 081 7683821
Email: lorenzo.peluso@unina.it

IPFIX Working Group
Internet-Draft
Intended status: BCP
Expires: December 13, 2012

B. Trammell
ETH Zurich
B. Claise
Cisco Systems, Inc.
June 11, 2012

Guidelines for Authors and Reviewers of IPFIX Information Elements
draft-ietf-ipfix-ie-doctors-03.txt

Abstract

This document provides guidelines for the definition of IPFIX Information Elements for addition to the IANA IPFIX Information Element registry, in order to extend the applicability of the IPFIX protocol to new operations and management areas.

Status of this Memo

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

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

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

This Internet-Draft will expire on December 13, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Intended Audience and Usage	3
1.2. Overview of relevant IPFIX documents	4
2. Terminology	4
3. How to apply IPFIX	5
4. Defining new Information Elements	6
4.1. Information Element naming	7
4.2. Information Element data types	7
4.3. Information Element numbering	8
4.4. Ancillary Information Element properties	9
4.5. Internal structure in Information Elements	9
4.6. Information Element multiplicity	10
4.7. Enumerated Values and Subregistries	11
4.8. Reversibility as per RFC 5103	11
4.9. Promotion of Enterprise-Specific Information Elements	11
4.10. Avoiding Bad Ideas in Information Element Design	12
5. The Information Element Lifecycle	13
5.1. The IE-DOCTORS process	13
5.2. Revising Information Elements	14
5.3. Deprecating Information Elements	15
5.4. Versioning the entire IANA Registry	16
6. When not to define new Information Elements	16
6.1. Maximizing reuse of existing Information Elements	16
6.2. Applying enterprise-specific Information Elements	18
7. Information Element Definition Checklist	18
8. Applying IPFIX to non-Flow Applications	21
9. Writing Internet-Drafts for IPFIX Applications	21
9.1. Example Information Element Definition	22
9.2. Defining Recommended Templates	23
10. A Textual Format for Specifying Information Elements and Templates	24
10.1. Information Element Specifiers	24
10.2. Specifying Templates	26
10.3. Specifying IPFIX Structured Data	27
11. Security Considerations	27
12. IANA Considerations	28
13. Acknowledgements	29
14. References	29
14.1. Normative References	29
14.2. Informative References	30
Appendix A. Example Information Element Definitions	31
A.1. sipResponseStatus	31
A.2. duplicatePacketDeltaCount	32
A.3. ambientTemperature	32
Authors' Addresses	33

1. Introduction

This document provides guidelines for the extension of the applicability of the IP Flow Information Export (IPFIX) protocol to network operations and management purposes outside the initial scope defined in "IPFIX Applicability Statement" [RFC5472]. These new applications are largely defined by creating new Information Elements beyond those in the IANA IPFIX Information Element Registry [iana-ipfix-assignments]. New applications may be further specified through additional RFCs defining and describing their usage.

We intend this document to enable the expansion of the applicability of IPFIX to new areas by experts in the working group or area directorate concerned with the technical details of the protocol or application to be measured or managed using IPFIX. This expansion would occur with the consultation of IPFIX experts informally called 'IE-Doctors'. It provides guidelines both for those defining new Information Elements as well as the IE-Doctors reviewing them.

1.1. Intended Audience and Usage

This document is meant for two separate audiences. For IETF contributors extending the applicability of IPFIX, it provides specifications and best practices to be used in deciding which Information Elements are necessary for a given existing or new application, defining these Information Elements, and deciding whether an RFC should be published to further describe the application. For the IPFIX experts appointed as IE-Doctors, and for IANA personnel changing the Information Element registry, it defines a set of acceptance criteria against which these proposed Information Elements should be evaluated.

This document is not intended to guide the extension of the IPFIX protocol itself, e.g. through new export mechanisms, data types, or the like; these activities should be pursued through the publication of standards-track RFCs by the IPFIX Working Group.

This document, together with [I-D.ietf-ipfix-information-model-rfc5102bis], defines the procedures for management of the IANA IPFIX Information Element Registry [iana-ipfix-assignments]. The practices outlined in this document are intended to guide experts when reviewing additions or changes to the Information Elements in the registry under Expert Review as defined in [RFC5226].

1.2. Overview of relevant IPFIX documents

[I-D.ietf-ipfix-protocol-rfc5101bis] defines the IPFIX Protocol, the IPFIX-specific terminology used by this document, and the data type encodings for each of the data types supported by IPFIX.

[I-D.ietf-ipfix-information-model-rfc5102bis] defines the basis of the IPFIX Information Model, referring to [iana-ipfix-assignments] for the specific Information Element definitions. It states that new Information Elements may be added to the Information Model on Expert Review basis, delegates the appointment of experts to an IESG Area Director, and refers to this document for details on the extension process. This document is intended to further codify the best practices to be followed by these experts, in order to improve the efficiency of this process.

[RFC5103] defines a method for exporting bidirectional flow information using IPFIX; this document should be followed when extending IPFIX to represent information about bidirectional network interactions in general. Additionally, new Information Elements should be annotated for their reversibility or lack thereof as per this document.

[RFC5610] defines a method for exporting information about Information Elements inline within IPFIX. In doing so, it explicitly defines a set of restrictions on the use of data types and semantics which are implied in [I-D.ietf-ipfix-protocol-rfc5101bis] and [I-D.ietf-ipfix-information-model-rfc5102bis]; these restrictions must be observed in the definition of new Information Elements, as in Section 4.4.

2. Terminology

Capitalized terms used in this document that are defined in the Terminology section of [I-D.ietf-ipfix-protocol-rfc5101bis] are to be interpreted as defined there.

An "application", as used in this document, refers to a candidate protocol, task, or domain to which IPFIX export, collection, and/or storage is applied, beyond those within the IPFIX Applicability statement [RFC5472]. By this definition, PSAMP [RFC5476] was the first new IPFIX application after the publication of the IPFIX protocol itself.

"IANA registry", as used in this document, unless otherwise noted, refers to the IANA IPFIX Information Element Registry [iana-ipfix-assignments].

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].

3. How to apply IPFIX

Though originally specified for the export of IP flow information, the message format, template mechanism, and data model specified by IPFIX lead to it being applicable to a wide variety of network management situations. In addition to flow information export, for which it was designed, and packet information export as specified by PSAMP [RFC5476], any application with the following characteristics is a good candidate for an IPFIX application:

- o The application's data flow is fundamentally unidirectional. IPFIX is a "push" protocol, supporting only the export of information from a sender (an Exporting Process) to a receiver (a Collecting Process). Request-response interactions are not supported by IPFIX.
- o The application handles discrete event information, or information to be periodically reported. IPFIX is particularly well suited to representing events, which can be scoped in time.
- o The application handles information about network entities. IPFIX's information model is network-oriented, so network management applications have many opportunities for information model reuse.
- o The application requires a small number of arrangements of data structures relative to the number of records it handles. The template-driven self-description mechanism used by IPFIX excels at handling large volumes of identically structured data, compared to representations which define structure inline with data (such as XML).

Most applications meeting these criteria can be supported over IPFIX. Once it's been determined that IPFIX is a good fit, the next step is determining which Information Elements are necessary to represent the information required by the application. Especially for network-centric applications, the IPFIX Information Element registry may already contain all the necessary Information Elements (see Section 6.1 for guidelines on maximizing Information Element reuse). In this case, no additional work within the IETF is necessary: simply define Templates and start exporting.

It is expected, however, that most applications will be able to reuse

some existing Information Elements, but may need to define some additional Information Elements to support all their requirements; in this case, see Section 4 for best practices to be followed in defining Information Elements.

Optionally, a Working Group or individual contributor may choose to publish an RFC detailing the new IPFIX application. Such an RFC should contain discussion of the new application, the Information Element definitions as in Section 4, as well as suggested Templates and examples of the use of those Templates within the new application as in Section 9.2. Section 10 defines a compact textual Information Element notation to be used in describing these suggested Templates and/or the use of IPFIX Structured Data [RFC6313] within the new application.

4. Defining new Information Elements

In many cases, a new application will require nothing more than a new Information Element or set of Information Elements to be exportable using IPFIX. An Information Element meeting the following criteria, as evaluated by appointed IPFIX experts, is eligible for inclusion in the Information Element registry:

- o The Information Element MUST be sufficiently unique within the registry. Its description MUST represent a substantially different meaning from that of any existing Information Element. A proposed Information Element which is a substantial duplicate of an existing Information Element is to be represented using the existing Information Element.
- o The Information Element SHOULD contain minimal internal structure; complex information should be represented with multiple simple Information Elements to be exported in parallel, as in Section 4.5.
- o The Information Element SHOULD be generally applicable to the application at hand, which SHOULD be of general interest to the community. Information Elements representing information about proprietary or nonstandard applications SHOULD be represented using enterprise-specific Information Elements as detailed in section 3.2 [RFC-EDITOR NOTE: verify section number] of [I-D.ietf-ipfix-protocol-rfc5101bis].

The definition of new Information Elements requires a descriptive name, a specification of the data type as one from the IPFIX Data Type Registry, and a human-readable description written in English. This section provides guidelines on each of these components of an

Information Element definition, referring to existing documentation such as [I-D.ietf-ipfix-information-model-rfc5102bis] as appropriate.

4.1. Information Element naming

Information Element Names should be defined in accordance with section 2.3 [RFC-EDITOR NOTE: verify section number] of [I-D.ietf-ipfix-information-model-rfc5102bis]; the most important naming conventions are repeated here for convenience.

- o Names of Information Elements SHOULD be descriptive.
- o Names of Information Elements MUST be unique within the registry.
- o Names of Information Elements MUST start with non-capitalized letters.
- o Composed names MUST use capital letters for the first letter of each component except for the first one. All other letters are non-capitalized, even for acronyms. Exceptions are made for acronyms containing non-capitalized letters, such as 'IPv4' and 'IPv6'. Examples are "sourceMacAddress" and "destinationIPv4Address."

In addition, new Information Elements pertaining to a specific protocol SHOULD name the protocol in the first word in order to ease searching by name (e.g. "sipMethod" for a SIP method, as would be used in a logging format for SIP based on IPFIX). Similarly, new Information Elements pertaining to a specific application SHOULD name the application in the first word.

4.2. Information Element data types

IPFIX provides a set of data types covering most primitives used in network measurement and management applications. The most appropriate data type should be chosen for the Information Element type, out of the IPFIX informationElementDataTypes subregistry at [iana-ipfix-assignments].

Information Elements representing an integral value with a natural width SHOULD be defined with the appropriate integral data type. This applies especially to values taken directly from fixed-width fields in a measured protocol. For example, tcpControlBits, the TCP flags byte, is an unsigned8, and tcpSequenceNumber is an unsigned32.

Information Elements representing counters or identifiers SHOULD be defined as signed64 or unsigned64, as appropriate, to maximize the range of values available; applications can to use reduced-size

encoding as defined in Section 6.2 [RFC-EDITOR NOTE: verify section number] of [I-D.ietf-ipfix-protocol-rfc5101bis] in cases where fewer than 2^{64} values are necessary.

Information Elements representing time values MUST be defined with appropriate precision. For example, a Information Element for a time measured at second-level precision should be defined as having a `dateTimeSeconds` data type, instead of `dateTimeMilliseconds`.

Information Elements of type `string` or `octetArray` which have length constraints (fixed length, minimum and/or maximum length) MUST note these constraints in their description.

The type of an Information Element MUST match the type of the data it represents. More specifically, information that could be represented as a `String`, but which better matches one of the other data types (e.g. an integral type for a number or enumerated type, an address type for an address) MUST be represented by the best-matching type, even if the data was represented using a different type in the source. In other words, an IPFIX application that exports Options Template Records mapping IP addresses to additional information about each host from an external database MUST use Information Elements of an address type to represent the addresses, even if the source database represented these as strings.

This document does not cover the addition of new Data Types or Data Type Semantics to the IPFIX Protocol. As such changes have important interoperability considerations and require implementation on both Collecting and Exporting Processes, they require a Standards Action as per [RFC5610]. However, note that the set of primitive types provided by IPFIX are applicable to most any appropriate application, so extending the type system is generally not necessary.

4.3. Information Element numbering

Each Information Element have a unique identifier in the IANA registry.

In general, when adding newly registered Information Elements to the registry, IANA SHOULD assign the lowest available Information Element identifier (the value column in [iana-ipfix-assignments]) in the range 128-32767, noting that prior noncontiguous allocation may lead to unassigned Information Elements with lower Information Element identifiers than some presently assigned Information Elements. This is the case with the PSAMP Information Model [RFC5477], which assigned a block of Information Elements identifiers starting at 300.

Information Element identifiers in the range 1-127 MUST NOT be

assigned unless the Information Element is compatible with the NetFlow V9 protocol as described in [RFC3954]. Such Information Elements may ONLY be requested by a NetFlow v9 expert, to be designated by the IESG to consult with IANA on NetFlow v9 compatibility with IPFIX.

4.4. Ancillary Information Element properties

Information Elements to which special semantics apply SHOULD define these semantics with one of the values in the Information Element Semantics registry, as described in Section 3.2 [RFC-EDITOR NOTE: verify section number] of [I-D.ietf-ipfix-information-model-rfc5102bis], subject to the restrictions given in Section 3.10 of [RFC5610]; in other words, the semantics and the type MUST be consistent.

When defining Information Elements representing a dimensioned quantity or entity count, the units of that quantity SHOULD be defined in the units field. This field takes its values from the IANA Information Element Units registry. If an Information Element expresses a quantity in units not yet in this registry, then the unit MUST be added to the Units registry at the same time the Information Element is added to the Information Element registry.

Additionally, when the range of values an Information Element can take is smaller than the range implied by its data type, the range SHOULD be defined within the Information Element registry.

4.5. Internal structure in Information Elements

The definition of Information Elements with internal structure with the structure defined in the Description field is NOT RECOMMENDED, except in the following cases:

- o The Information Element is a direct copy of a structured entity in a measured protocol (e.g. the tcpControlBits Information Element for the flags byte from the TCP header)
- o The Information Element represents a section of a packet of protocol entity, in raw form as captured from the wire (e.g. the mplsLabelStackSection Information Element for the MPLS label stack)
- o The Information Element represents a set of flags which are tightly semantically related, where representing the flags as separate one-byte booleans would be inefficient, and which should always appear together in a data record (e.g., the anonymizationFlags Information Element for specifying optional

features of anonymization techniques)

In other cases, candidate Information Elements with internal structure SHOULD be decomposed into multiple primitive Information Elements to be used in parallel. For more complicated semantics, where the structure is not identical from Data Record to Data Record, or where there is semantic dependency between multiple decomposed primitive Information Elements, use the IPFIX Structured Data [RFC6313] extension instead.

As an example of information element decomposition, consider an application-level identifier called an "endpoint", which represents a {host, port, protocol} tuple. Instead of allocating an opaque, structured "source endpoint" Information Element, the source endpoint should be represented by three separate Information Elements: "source address", "source port", "transport protocol". In this example, the required information elements already exist in the Information Element registry: sourceIPv4Address or sourceIPv6Address, sourceTransportPort, protocolIdentifier. Indeed, as well as being good practice, this normalization down to non-structured Information Elements also increases opportunities for reuse as in Section 6.1.

The decomposition of data with internal structure SHOULD avoid the definition of Information Elements with a meaning too specific to be generally useful, or that would result in either the export of meaningless data or a multitude of templates to handle different multiplicities. More information on multiplicities is given in the following section.

4.6. Information Element multiplicity

Some Information Elements may represent information with a multiplicity other than one; i.e., items that may occur multiple times within the data to be represented in a single IPFIX record. In this case, there are several options, depending on the circumstances:

- o As specified in section 8 [RFC-EDITOR NOTE: verify section number] of [I-D.ietf-ipfix-protocol-rfc5101bis]: "if an Information Element is required more than once in a Template, the different occurrences of this Information Element SHOULD follow the logical order of their treatments by the Metering Process." In other words, in cases where the items have a natural order (e.g., the order in which they occur in the packet), and the multiplicity is the same for each record, the information can be modeled by containing multiple instances of the Information Element representing a single item within the Template Record describing the Data Records.

- o In cases where the items have a variable multiplicity, a basicList of the Information Element representing a single item can be used as in the IPFIX Structured Data [RFC6313] extension.
- o If the multiple-item structure is taken directly from bytes observed on the wire by the Metering Process or otherwise taken from the application being measured, the multiple-item structure can be exported as a variable-length octetArray Information Element holding the raw content.

Specifically, new Information Element SHOULD NOT encode any multiplicity or ordinality information into the definition of the Information Element itself.

4.7. Enumerated Values and Subregistries

When defining an Information Element that takes an enumerated value from a set of values which may change in the future, this enumeration MUST be defined by an IANA registry or subregistry. For situations where an existing registry defines the enumeration (e.g., the IANA Protocol Numbers registry for the protocolIdentifier Information Element), that registry MUST be used. Otherwise, a new IPFIX subregistry MUST be defined for the enumerated value, to be modified subject to Expert Review [RFC5226].

4.8. Reversibility as per RFC 5103

[RFC5103] defines a method for exporting bidirectional flows using a special Private Enterprise Number to define reverse-direction variants of IANA Information Elements, and a set of criteria for determining whether an Information Element may be reversed using this method. Since almost all Information Elements are reversible, [RFC5103] enumerates those which Information Elements which were defined at the time of its publication which are NOT reversible.

New non-reversible Information Elements SHOULD contain a note in the description stating that they are not reversible.

4.9. Promotion of Enterprise-Specific Information Elements

Some Information Elements may start their lifecycle outside the IANA registry as enterprise-specific Information Elements scoped to a Private Enterprise Number. One stated goal of enterprise-specific Information Elements is pre-standards product delivery and experimentation; should these experiments be successful and the Information Elements generally useful, these SHOULD subsequently be registered with IANA.

In order to support transition from experimental registration to IANA registration, the IANA registry provides an optional "enterprise-specific IE reference" column for each Information Element. In cases of promoted enterprise-specific Information Elements, this column in the registry MAY contain the private enterprise and Information Element numbers of the enterprise-specific version(s) of the Information Element.

4.10. Avoiding Bad Ideas in Information Element Design

In general, the existence of a similarly-defined Information Element in the IANA registry sets a precedent which may be followed to determine whether a given proposed Information Element "fits" within the registry. Indeed, the rules specified by this document could be interpreted to mean "make new Information Elements that look like existing Information Elements". However, for reasons of history, there are several Information Elements within the IANA registry which do not follow best practices in Information Element design. These Information Elements are not necessarily so flawed so as to require deprecation, but they should be explicitly ignored when looking for guidance as to whether a new Information Element should be added.

Before registering a new Information Element, it must be determined that it would be sufficiently unique within the registry. This evaluation has not always been done in the past, and the existence of the Information Elements defined without this evaluation should not be taken as an example that such Information Element definition practices should be followed in the future. Specific examples of such Information Elements include `initiatorOctets` and `responderOctets` (which duplicate `octetDeltaCount` and its reverse per [RFC5103]) and `initiatorPackets` and `responderPackets` (the same, for `packetDeltaCount`).

As mentioned in Section 4.2, the type of an Information Element SHOULD match the type of data the Information Element represents. An example of how not to do this is presented by the `p2pTechnology`, `tunnelTechnology`, and `encryptedTechnology` Information Elements: these represent a three-state enumeration using a String. The example set by these Information Elements SHOULD NOT be followed in the definition of new Information Elements.

As mentioned in Section 4.6, an Information Element definition SHOULD NOT include any ordinality or multiplicity information. The only example of this within the IANA registry the following list of assigned IPFIX Information Elements: `mplsTopLabelStackSection`, `mplsLabelStackSection2`, `mplsLabelStackSection3`, `mplsLabelStackSection4`, `mplsLabelStackSection5`, `mplsLabelStackSection6` `mplsLabelStackSection7`,

mplsLabelStackSection8, mplsLabelStackSection9, and mplsLabelStackSection10. The only distinction between those almost-identical Information Elements is the position within the MPLS stack. This Information Element design pattern met an early requirement of the definition of IPFIX which was not carried forward into the final specification -- namely, that no semantic dependency was allowed between Information Elements in the same Record -- and as such SHOULD NOT be followed in the definition of new Information Elements. In this case, since the size of the MPLS stack will vary from flow to flow, it should be exported using IPFIX Structured Data [RFC6313] where supported, as a basicList of MPLS label entries, or as a raw MPLS label stack using the variable-length mplsLabelStackSection Information Element.

5. The Information Element Lifecycle

Once an Information Element or set of Information Elements has been identified for a given application, Information Element specifications in accordance with Section 4 are submitted to IANA to follow the IE-DOCTORS process, as defined below. This process is also used for other changes to the registry, such as deprecation or revision, as described later in this section.

5.1. The IE-DOCTORS process

Requests to change the IANA Information Element registry or a linked subregistry are submitted to IANA, which forwards the request to a designated group of experts (IE-DOCTORS) appointed by the IESG. This group of experts reviews the request for such things as compliance with this document, compliance with other applicable IPFIX-related RFCs, and consistency with the currently defined set of Information Elements.

Authors are expected to review compliance with the specifications in this document to check their submissions before sending them to IANA.

IE-DOCTORS reviewers should endeavor to complete referred reviews in a timely manner. If the request is acceptable, the IE-DOCTORS signify their approval to IANA, which changes the IANA Information Element registry. If the request is not acceptable, the IE-DOCTORS can coordinate with the requestor to change the request to be compliant. The IE-DOCTORS may also choose in exceptional circumstances to reject clearly frivolous or inappropriate change requests outright.

5.2. Revising Information Elements

The Information Element status field in the Information Element Registry is defined in [I-D.ietf-ipfix-information-model-rfc5102bis] to allow Information Elements to be 'current', 'deprecated' or 'obsolete'. No Information Elements are as of this writing deprecated or obsolete, and [I-D.ietf-ipfix-information-model-rfc5102bis] does not define any policy for using them. Additionally, no policy is defined for revising Information Element registry entries or addressing errors therein. To be certain, changes and deprecations within the Information Element registry are not encouraged, and should be avoided to the extent possible. However, in recognition that change is inevitable, this section is intended to remedy this situation.

The primary requirement in the definition of a policy for managing changes to existing Information Elements is avoidance of interoperability problems; IPFIX experts appointed to review changes to the Information Element Registry MUST work to maintain interoperability above all else. Changes to Information Elements already in use may only be done in an interoperable way; necessary changes which cannot be done in a way to allow interoperability with unchanged implementations MUST result in deprecation.

A change to an Information Element is held to be interoperable only when:

- o it involves the correction of an error which is obviously only editorial; or
- o it corrects an ambiguity in the Information Element's definition, which itself leads to non-interoperability severe enough to prevent the Information Element's usage as originally defined (e.g., a prior change to `ipv6ExtensionHeaders`); or
- o it expands the Information Element's data type without changing how it is represented (e.g., changing `unsigned32` to `unsigned64`, as with a prior change to `selectorId`); or
- o it defines a previously undefined or reserved enumerated value, or one or more previously reserved bits in an Information Element with flag semantics; or
- o it expands the set of permissible values in the Information Element's range; or
- o it harmonizes with an external reference which was itself corrected.

If a change is permissible, it is sent to IANA, which passes it to the appointed experts for review; if there is no objection to the change from any appointed expert, IANA makes the change in the Information Element Registry. The requestor of the change is appended to the Requestor in the registry.

Each Information Element in the IANA registry has a revision number, starting at zero. Each change to an Information Element following this process increments the revision number by one. Since any revision must be interoperable according to the criteria above, there is no need for the IANA registry to store information about old revisions.

5.3. Deprecating Information Elements

Changes that are not permissible by these criteria may only be handled by deprecation. An Information Element MAY be deprecated and replaced when:

- o the Information Element definition has an error or shortcoming which cannot be permissibly changed as above; or
- o the deprecation harmonizes with an external reference which was itself deprecated through that reference's accepted deprecation method; or
- o changes in the IPFIX Protocol or its extensions, or in community understanding thereof, allow the information represented by the Information Element to be represented in a more efficient or convenient way. Deprecation in this circumstance additionally requires the assent of the IPFIX Working Group, and should be specified in the Internet Draft(s) defining the protocol change.

A request for deprecation is sent to IANA, which passes it to the IE-DOCTORS for review, as above. When deprecating an Information Element, the Information Element description MUST be updated to explain the deprecation, as well as to refer to any new Information Elements created to replace the deprecated Information Element. The revision number of an Information Element is incremented upon deprecation.

Deprecated Information Elements SHOULD continue to be supported by Collecting Processes, but SHOULD NOT be exported by Exporting Processes. The use of deprecated Information Elements SHOULD result in a log entry or human-readable warning at the Exporting and Collecting Processes. After a period of time determined in the eyes of the IE-DOCTORS experts to be reasonable in order to allow deployed Exporting Processes to be updated to account for the deprecation, a

deprecated Information Element may be made obsolete. Obsolete Information Elements MUST NOT be supported by either Exporting or Collecting Processes. The receipt of obsolete Information Elements SHOULD be logged by the Collecting Process.

Names of deprecated or obsolete Information Elements MUST NOT be reused.

5.4. Versioning the entire IANA Registry

Consider a typical Collector implementation, which regularly downloads the entire registry in order to be compliant with the latest of set of supported IEs. While a registry revision number might seem advantageous for the Collector at first glance (avoiding the one by one comparison of all IE revisions), it is not necessary, as the IPFIX IANA registry specifies the date at which the registry was last updated in the "Last Updated" field. For purposes of identifying the latest set of Information Element versions specified in registry, the last revision date of the Information Element registry (available in the registry XML source, or from the Last-Modified: header of [iana-ipfix-assignments]) SHOULD be used.

6. When not to define new Information Elements

Due to the limited number space of Information Elements in the IANA registry, avoiding redundancy and clutter in the Information Element registry is important in defining new applications. New Information Elements SHOULD NOT be added to the Information Element registry unless there is an intent to implement and deploy applications using them; research or experimental applications SHOULD use enterprise-specific Information Elements as in Section 6.2 instead.

The subsections below provide guidelines for reuse of existing Information Elements, as well as guidelines on using enterprise-specific Information Elements instead of adding Information Elements in the registry.

6.1. Maximizing reuse of existing Information Elements

Whenever possible, new applications should prefer usage of existing IPFIX Information Elements to the creation of new Information Elements. IPFIX already provides Information Elements for every common Layer 4 and Layer 3 packet header field in the IETF protocol suite, basic Layer 2 information, basic counters, timestamps and time ranges, and so on. When defining a new Information Element similar to an existing one, reviewers shall ensure that the existing one is not applicable.

Note that this guideline to maximize reuse does not imply that an Information Element that represents the same information from a packet as a existing Information Element should not be added to the registry. For example, consider the `ipClassOfService` (Element ID 5), `ipDiffServCodePoint` (Element ID 98), and `ipPrecedence` (Element ID 196) Information Elements. These all represent subsets of the same field in an IP version 4 packet header, but different uses of these bits. The representation in one or another of these Information Elements contains information in itself as to how the bits were interpreted by the Metering Process.

On the other hand, simply changing the context in which an Information Element will be used is insufficient reason for the definition of a new Information Element. For example, an extension of IPFIX to log detailed information about HTTP transactions alongside network-level information should not define `httpClientAddress` and `httpServerAddress` Information Elements, preferring instead the use of `sourceIPv[46]Address` and `destinationIPv[46]Address`.

Applications dealing with bidirectional interactions should use Bidirectional Flow Support for IPFIX [RFC5103] to represent these interactions.

Existing timestamp and time range Information Elements should be reused for any situation requiring simple time stamping of an event: for single observations, the `observationTime*` Information Elements from PSAMP are provided, and for events with a duration, the `flowStart*` and `flowEnd*` Information Elements suffice. This arrangement allows minimal generic time handling by existing Collecting Processes and analysis workflows. New timestamp Information Elements should ONLY be defined for semantically distinct timing information (e.g., an IPFIX-exported record containing information about an event to be scheduled in the future).

In all cases the use of absolute timestamp Information Elements (e.g. `flowStartMilliseconds`) is RECOMMENDED, as these Information Elements allow for maximum flexibility in processing with minimal overhead. Timestamps based on the export time header in the enclosing IPFIX Message (e.g. `flowStartTimeDeltaMicroseconds`) MAY be used if high-precision timing is important, export bandwidth or storage space is limited, timestamps comprise a relatively large fraction of record size, and the application naturally groups records into IPFIX Messages. Timestamps based on information which must be exported in a separate Data Record defined by an Options Template (e.g. `flowStartSysUpTime`) MAY be used only in the context of an existing practice of using runtime-defined epochs for the given application. New applications SHOULD avoid these structures when possible.

6.2. Applying enterprise-specific Information Elements

IPFIX provides a mechanism for defining enterprise-specific Information Elements, as in Section 3.2 [RFC-EDITOR NOTE: verify section number] of [I-D.ietf-ipfix-protocol-rfc5101bis]. These are scoped to a vendor's or organization's Structure of Management Information (SMI) Private Enterprise Number, and are under complete control of the organization assigning them.

For situations in which interoperability is unimportant, new information SHOULD be exported using enterprise-specific Information Elements instead of adding new Information Elements to the registry. These situations include:

- o export of implementation-specific information, or
- o export of information derived in a commercially-sensitive or proprietary method, or
- o export of information or meta-information specific to a commercially-sensitive or proprietary application.

While work within the IETF generally does not fall into these categories, enterprise-specific Information Elements are also useful for pre-standardization testing of a new IPFIX application. While performing initial development and interoperability testing of a new application, the Information Elements used by the application SHOULD NOT be submitted to IANA for inclusion in the registry. Instead, these experimental Information Elements SHOULD be represented as enterprise-specific until their definitions are finalized, then transitioned from enterprise-specific to IANA-defined upon finalization. To support this transition, the IANA registry provides an experimental IE reference as defined in Section 4.9.

7. Information Element Definition Checklist

The following three checklists, condensed from the rest of this document, can be used when defining and reviewing Information Elements; they refer back to the section of this document from which they are taken. These checklists are intended for the definition of new Information Elements; revision should follow the process defined in Section 5.2, and deprecation should follow the process defined in Section 5.3.

Though many of the considerations in this document require the subjective judgement of Information Element authors, reviewers, and IANA, certain parts of the process may be made simpler through tool

support. Items on these checklists which could be easily automated or assisted by tools are annotated with "(tool support)". Other items on these checklists require some level of subjective judgement; checks for semantic uniqueness may additionally be supported by textual analysis of descriptions in the future.

Checklist 1 contains conditions which MUST be met by all proposed Information Elements:

- o The name MUST be unique within the registry, and not reuse the name of any current, deprecated, or obsolete Information Element. (Section 4.1) (tool support)
- o The description MUST be sufficiently semantically unique within the registry, representing a substantially different meaning from any current, deprecated, or obsolete Information Element. (Section 4)
- o The name MUST start with a non-capitalized letter. (Section 4.1) (tool support)
- o Names composed of more than one word MUST use capital letters for the first letter of each component except for the first one; all other letters are non-capitalized, even for acronyms. Exceptions are made for acronyms containing non-capitalized letter, such as 'IPv4' and 'IPv6'. (Section 4.1) (tool support)
- o The data type MUST match the type of the data being represented. (Section 4.2)
- o Data type semantics MUST be appropriate for the data type. (Section 4.4) (tool support)
- o The Information Element identifier assigned by IANA MUST be unique. (Section 4.3) (tool support)
- o The Information Element identifier assigned by IANA MUST NOT be in the range 1-127 unless it is compatible with the counterpart Information Element used in NetFlow V9 [RFC3954]. (Section 4.3)

Checklist 2 contains conditions which MUST be met by proposed Information Elements with certain properties, as noted:

- o Time values MUST be defined with appropriate precision. (Section 4.2)
- o Strings and octet arrays with length restrictions MUST note those length restrictions in their descriptions. (Section 4.2)

- o Enumerations MUST refer to an IANA registry or subregistry. If no suitable registry or subregistry exists, a new subregistry of the IPFIX Information Element registry MUST be created for the enumeration, to be modified subject to Expert Review [RFC5226]. (Section 4.7)

Checklist 3 contains conditions which SHOULD be met by proposed Information Elements:

- o The name of an Information Element pertaining to a specific protocol SHOULD contain the name of the protocol as the first word. (Section 4.1)
- o The name of an Information Element pertaining to a specific application SHOULD contain the name of the application as the first word. (Section 4.1)
- o Information Elements representing integral values SHOULD use a data type for the appropriate width for the value. (Section 4.2)
- o Information Elements representing counters or identifiers SHOULD be represented as signed64 or unsigned64, as appropriate. (Section 4.2)
- o An Information Element SHOULD NOT contain internal structure, subject to the exceptions in Section 4.5; candidate Information Elements with internal structure SHOULD be decomposed into multiple Information Elements. (Section 4.5)
- o An Information Element SHOULD NOT contain multiplicity or ordinality information within the definition of the Information Element itself. (Section 4.6)
- o Data type semantics SHOULD be defined, if appropriate. (Section 4.4) (tool support)
- o Units SHOULD be defined, if appropriate, with new units added to the Information Element Units subregistry if necessary. (Section 4.4) (tool support)
- o Ranges SHOULD be defined, if appropriate. (Section 4.4) (tool support)
- o Non-reversible Information Elements (see [RFC5103]) SHOULD note non-reversibility in the description. (Section 4.8)
- o Information Elements created to replace enterprise-specific Information Elements used for experimental or testing purposes

SHOULD contain a reference to the enterprise-specific Information Element they replace. (Section 4.9)

- o The Information Element Identifier assigned by IANA for IEs not compatible with their counterparts in NetFlow V9 SHOULD be the lowest available identifier above 128. (Section 4.3) (tool support)
- o Information Elements to be registered with IANA SHOULD BE intended for implementation and deployment on production networks.

8. Applying IPFIX to non-Flow Applications

At the core of IPFIX is its definition of a Flow, a set of packets sharing some common properties crossing an observation point within a certain time window. However, the reliance on this definition does not preclude the application of IPFIX to domains which are not obviously handling flow data according to it. Most network management data collection tasks, those to which IPFIX is most applicable, have at their core the movement of packets from one place to another; by a liberal interpretation of the common properties defining the flow, then, almost any event handled by these can be held to concern data records conforming to the IPFIX definition of a Flow.

Non-flow information defining associations or key-value pairs, on the other hand, are defined by IPFIX Options Templates. Here, the Information Elements within an Options Template Record are divided into Scope Information Elements which define the key, and non-scope Information Elements which define the values associated with that key. Unlike Flows, Data Records defined by Options Template are not necessarily scoped in time; these Data Records are generally held to be in effect until a new set of values for a specific set of keys is exported. While this mechanism is often used by IPFIX to export metadata about the collection infrastructure, it is applicable to any association information.

An IPFIX application can mix Data Records described either type of template in an IPFIX Message or Message stream, and exploit relationships among the Flow Keys, values, and Scopes to create interrelated data structures. See [RFC5473] for an example application of this.

9. Writing Internet-Drafts for IPFIX Applications

When a new application is complex enough to require additional

clarification or specification as to the use of the defined Information Elements, this may be given in an Internet-Draft. Internet-Drafts for new IPFIX applications are best submitted to a Working Group with expertise in the area of the new application, or as independent submissions.

When defining new Information Elements in an Internet-Draft, the Internet-Draft SHOULD contain a section (or subsection) for each Information Element, which contains the attributes in Section 4 in human-readable form. An example subsection is given below. These Information Element descriptions SHOULD NOT assign Information Element numbers, instead using placeholder identifiers for these numbers (e.g. "AAA", "BBB", "CCC", or "TBD1", "TBD2", "TBD3") and a note to IANA in the IANA Considerations section to replace those placeholders in the document with Information Element numbers when the numbers are assigned. The use of these placeholder definitions allows references to the numbers in e.g. box-and-line diagrams or template definitions as in Section 10.

9.1. Example Information Element Definition

This is an example of an Information Element definition which would appear in an Internet-Draft. The name appears in the section title.

Description: Description goes here.

Data Type: Data type goes here; obligatory

Data Type Semantics: Data type semantics, if any, go here; optional

Units: Units, if any, go here; optional

Range: Range, if not implied by the data type, goes here; optional

References: References to other RFCs or documents outside the IETF, in which additional information is given, or which are referenced by the description, go here; optional

ElementId: ElementId, if known, or TBD to be filled in by IANA at publication time.

Replaces Enterprise-Specific Element: If the Information Element replaces an existing enterprise-specific Information Element, the PEN and Information Element number go here, separated by a slash (e.g. 35566/123) as in Section 10.

9.2. Defining Recommended Templates

New IPFIX applications SHOULD NOT, in the general case, define fixed templates for export, as this throws away much of the flexibility afforded by IPFIX. However, fixed template export is permissible in the case that the export implementation must operate in a resource constrained environment, and/or that the application is replacing an existing fixed-format binary export format in a maximally compatible way. In any case, Collecting Processes for such applications SHOULD support reordered Templates or Templates with additional Information Elements.

An Internet-Draft clarifying the use of new Information Elements SHOULD include any recommended Template or Options Template Records necessary for supporting the application, as well as examples of records exported using these Template Records. In defining these Template Records, such Internet-Drafts SHOULD mention, subject to rare exceptions as above:

- o that the order of Information Elements within a Template is not significant;
- o that Templates on the wire for the application may also contain additional Information Elements beyond those specified in the recommended Template;
- o that a stream of IPFIX Messages supporting the application may also contain Data Records not described by the recommended Templates; and
- o that any reader of IPFIX Messages supporting the application MUST accept these conditions.

Definitions of recommended Template Records for flow-like information, where the Flow Key is well-defined, SHOULD indicate which of the Information Elements in the recommended Template are Flow Keys.

Recommended Templates are defined, for example, in [RFC5476] for PSAMP packet reports (section 6.4) and extended packet reports (section 6.5). Recommended Options Templates are defined extensively throughout the IPFIX documents, including in the protocol document itself [I-D.ietf-ipfix-protocol-rfc5101bis] for exporting export statistics; in the file format [RFC5655] for exporting file metadata; and in Mediator intermediate process definitions such as [RFC6235] for intermediate process metadata. The discussion in these examples is a good model for recommended template definitions.

10. A Textual Format for Specifying Information Elements and Templates

Example Templates given in existing IPFIX documents are generally expressed using bitmap diagrams of the respective Templates. These are illustrative of the wire representation of simple Templates, but not particularly readable for more complicated recommended Templates, provide no support for rapid implementation of new Templates, and do not adequately convey the optional nature of ordering and additional Information Elements as above. Therefore, we define a RECOMMENDED textual format for specifying Information Elements and Templates in Internet-Drafts in this section.

Here we define a simple textual syntax for describing IPFIX Information Elements and IPFIX Templates, with human readability, human writability, compactness, and ease of parser/generator implementation without requiring external XML support as design goals. It is intended both for use in human communication (e.g., in new Internet-Drafts containing higher-level descriptions of IPFIX Templates, or describing sets of new IPFIX Information Elements for supporting new applications of the protocol) as well as at runtime by IPFIX implementations.

10.1. Information Element Specifiers

The basis of this format is the textual Information Element Specifier, or IESpec. An IESpec contains each of the four important aspects of an Information Element: its name, its number, its type, and its size, separated by simple markup based on various types of brackets. Fully-qualified IESpecs may be used to specify existing or new Information Elements within an Information Model, while either fully-qualified or partial IESpecs may be used to define fields in a Template.

Bare words are used for Information Element names, and each aspect of information associated with an Information Element is associated with a type of brackets:

- o `()` parentheses for Information Element numbers,
- o `<>` angles for Information Element data types, and
- o `[]` square brackets for Information Element sizes.
- o `{ }` curly braces contain an optional space-separated list of context identifiers to be associated with an Information Element, as described in more detail in Section 10.2

The symbol `+` is reserved for Information Element nesting within

structured data elements; these are described in and Section 10.3, respectively.

Whitespace in IESpecs is insignificant; spaces can be added after each element in order, e.g., to align columns for better readability.

The basic form of a fully-qualified IESpec for an IANA-registered Information Element is as follows:

```
name(number)<type>[size]
```

where 'name' is the name of the Information Element in UTF-8, 'number' is the Information Element as a decimal integer, 'type' is the name of the data type as in the IANA informationElementDataTypes registry, and 'size' is the length of the Information Element in octets as a decimal integer, where 65535 or the string 'v' signifies a variable-length Information Element. [size] may be omitted; in this case, the data type's native or default size is assumed.

The basic form of a fully-qualified IESpec for an enterprise-specific Information Element is as follows:

```
name(pen/number)<type>[size]
```

where 'pen' is the Private Enterprise Number as a decimal integer.

A fully-qualified IESpec is intended to express enough information about an Information Element to decode and display Data Records defined by Templates containing that Information Element. Range, unit, semantic, and description information, as in [RFC5610], is not supported by this syntax.

Example fully-qualified IESpecs follow:

```
octetDeltaCount(1)<unsigned64>[8]
```

```
octetDeltaCount(1)<unsigned64> (unsigned64 is natively 8 octets  
long)
```

```
sourceIPv4Address(8)<ipv4Address>
```

```
wlanSSID(146)<string>[v]
```

```
sipRequestURI(35566/403)<string>[65535]
```

A partial IESpec is any IESpec that is not fully-qualified; these are useful when defining templates. A partial IESpec is assumed to take missing values from its canonical definition, for example, the IANA

registry. At minimum, a partial IESpec must contain a name, or a number. Any name, number, or type information given with a partial IESpec must match the values given in the Information Model; however, size information in a partial IESpec overrides size information in the Information Model; in this way, IESpecs can be used to express reduced-length encoding for Information Elements.

Example partial IESpecs follow:

- o `octetDeltaCount`
- o `octetDeltaCount[4]` (reduced-length encoding)
- o `(1)`
- o `(1)[4]` (reduced length encoding; note that this is exactly equivalent to an Information Element specifier in a Template)

10.2. Specifying Templates

A Template can then be defined simply as an ordered, newline-separated sequence of IESpecs. IESpecs in example Templates illustrating a new application of IPFIX SHOULD be fully-qualified. Flow Keys may be optionally annotated by appending the {key} context to the end of each Flow Key specifier. A template counting packets and octets per five-tuple with millisecond precision in IESpec syntax is shown below.

```
flowStartMilliseconds(152)<dateTimeMilliseconds>[8]  
flowEndMilliseconds(153)<dateTimeMilliseconds>[8]  
octetDeltaCount(1)<unsigned64>[8]  
packetDeltaCount(2)<unsigned64>[8]  
sourceIPv4Address(8)<ipv4Address>[4]{key}  
destinationIPv4Address(12)<ipv4Address>[4]{key}  
sourceTransportPort(7)<unsigned16>[2]{key}  
destinationTransportPort(11)<unsigned16>[2]{key}  
protocolIdentifier(4)<unsigned8>[1]{key}
```

Sample flow template in IESpec syntax

An Options Template is specified similarly. Scope is specified appending the {scope} context to the end of each IESpec for a Scope IE. Due to the way Information Elements are represented in Options Templates, all {scope} IESpecs must appear before any non-scope IESpec. The Flow Key Options Template defined in section 4.4 [RFC-EDITOR NOTE: verify section number] of [I-D.ietf-ipfix-protocol-rfc5101bis] in IESpec syntax is shown below:


```
templateId(145)<unsigned16>[2]{scope}  
flowKeyIndicator(173)<unsigned64>[8]
```

Flow Key Options Template in IESpec syntax

10.3. Specifying IPFIX Structured Data

IESpecs can also be used to illustrate the structure of the information exported using the IPFIX Structured Data extension [RFC6313]. Here, the semantics of the structured data elements are specified using contexts, and the information elements within each structured data element follow the structured data element, prefixed with + to show they are contained therein. Arbitrary nesting of structured data elements is possible by using multiple + signs in the prefix. For example, a basic list of IP addresses with "one or more" semantics would be expressed using partially qualified IESpecs as follows:

```
basicList{oneOrMoreOf}  
+sourceIPv4Address(8)[4]
```

Sample basicList in IESpec syntax

And an example subTemplateList itself containing a basicList is shown below:

```
subTemplateList{allOf}  
+basicList{oneOrMoreOf}  
++sourceIPv4Address(8)[4]  
+destinationIPv4Address(12)[4]
```

Sample subTemplateList in IESpec syntax

This describes a subTemplateMultilist containing all of the expressed set of source-destination pairs, where the source address itself could be one of any number in a basicList (e.g., in the case of SCTP multihoming).

The contexts associable with structured data Information Elements are the semantics, as defined in section 4.4 of [RFC6313]; a structured data Information Element without any context is taken to have undefined semantics. More information on the application of structured data is available in [RFC6313].

11. Security Considerations

The security aspects of new Information Elements must be considered

in order not to give a potential attacker too much information. For example, the "A Framework for Packet Selection and Reporting" [RFC5474] concluded in section 12.3.2 that the hash functions private parameters should not be exported within IPFIX.

If some security considerations are specific to an Information Element, they MUST be mentioned in the Information Element description. For example, the `ipHeaderPacketSection` in the IPFIX registry mentions: "This Information Element, which may have a variable length, carries a series of octets from the start of the IP header of a sampled packet. With sufficient length, this element also reports octets from the IP payload, subject to [RFC2804]. See the Security Considerations section."

These security considerations SHOULD also be stressed in an accompanying Internet-Draft, as in Section 9. For example, the "Packet Sampling (PSAMP) Protocols Specification" [RFC5476] specifies: "In the basic Packet Report, a PSAMP Device exports some number of contiguous bytes from the start of the packet, including the packet header (which includes link layer, network layer and other encapsulation headers) and some subsequent bytes of the packet payload. The PSAMP Device SHOULD NOT export the full payload of conversations, as this would mean wiretapping [RFC2804]. The PSAMP Device MUST respect local privacy laws."

Internet Drafts which define Information Elements SHOULD list and discuss any security impact of those Information Elements in their Security Considerations sections.

12. IANA Considerations

With respect to the management of the IPFIX Information Element registry and associated subregistries located at [iana-ipfix-assignments], this document defines a process for IANA in Section 5.1, and includes a set of guidelines for IANA for applying this process in Section 4, Section 5, and Section 6.

The IE-DOCTORS experts and the NetFlow V9 expert mentioned within this document are to be appointed by the IESG.

In addition, in order to support more effective management of the Information Element lifecycle as defined in Section 5, this document specifies the addition of three new columns for the registry:

Revision: a serial revision number for each Information Element, beginning at 0 for all presently existing and newly created Information Elements. This column is to be managed by IANA in order to support the process defined in Section 5.2, and need not be present in any Information Element definition.

Date: the date at which the Information Element was created or last modified. This column is to be managed by IANA in order to support the process defined in Section 5.2, and need not be present in any Information Element definition.

Enterprise-specific reference: for Information Elements which were deployed as enterprise-specific Information Elements for experimentation and testing, and subsequently registered in the IANA registry, specifies the private enterprise number (PEN)/IE number pair(s) of the equivalent experimental Information Element(s). Note that this column may contain more than one entry per Information Element. This column SHOULD be present in any Information Element definition referencing an enterprise-specific Information Element, as in Section 4.9.

[IANA NOTE: Note that the examples in Appendix A are NOT to be added to the IPFIX Information Element registry upon the publication of this document.]

13. Acknowledgements

Thanks to Paul Aitken, Andrew Feren, and Dan Romascanu for their valuable reviews and feedback. This work is materially supported by the European Union Seventh Framework Programme under grant agreement 257315 (DEMONS).

14. References

14.1. Normative References

[I-D.ietf-ipfix-protocol-rfc5101bis]

Claise, B. and B. Trammell, "Specification of the IP Flow Information eXport (IPFIX) Protocol for the Exchange of Flow Information", draft-ietf-ipfix-protocol-rfc5101bis-01 (work in progress), March 2012.

[I-D.ietf-ipfix-information-model-rfc5102bis]

Claise, B. and B. Trammell, "Information Model for IP Flow Information eXport (IPFIX)", draft-ietf-ipfix-information-model-rfc5102bis-01 (work in

progress), March 2012.

- [RFC5103] Trammell, B. and E. Boschi, "Bidirectional Flow Export Using IP Flow Information Export (IPFIX)", RFC 5103, January 2008.
- [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.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.
- [RFC6313] Claise, B., Dhandapani, G., Aitken, P., and S. Yates, "Export of Structured Data in IP Flow Information Export (IPFIX)", RFC 6313, July 2011.

14.2. Informative References

- [RFC2804] IAB and IESG, "IETF Policy on Wiretapping", RFC 2804, May 2000.
- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.
- [RFC3954] Claise, B., "Cisco Systems NetFlow Services Export Version 9", RFC 3954, October 2004.
- [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.
- [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.
- [RFC5560] Uijterwaal, H., "A One-Way Packet Duplication Metric", RFC 5560, May 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.
- [RFC6235] Boschi, E. and B. Trammell, "IP Flow Anonymization Support", RFC 6235, May 2011.
- [iana-ipfix-assignments]
Internet Assigned Numbers Authority, "IP Flow Information Export Information Elements
(<http://www.iana.org/assignments/ipfix/ipfix.xml>)".

Appendix A. Example Information Element Definitions

This section contains a few example Information Element definitions as they would appear in an Internet-Draft. Note the conformance of these examples to the guidelines in Section 4.

The sipResponseStatus Information Element (Appendix A.1) illustrates the addition of an Information Element representing Layer 7 application information, with a reference to the registry containing the allowable values. The duplicatePacketDeltaCount Information Element (Appendix A.2) illustrates the addition of a new metric, with a reference to the RFC defining the metric. The ambientTemperature Information Element (Appendix A.3) illustrates the addition of a new measured value outside the area of traditional networking applications.

A.1. sipResponseStatus

Description: The SIP Response code as an integer, as in the Response Codes registry at <http://www.iana.org/assignments/sip-parameters> defined in [RFC3261] and amended in subsequent RFCs. The presence of this Information Element in a SIP Message record marks it as describing a SIP response; if absent, the record describes a SIP request.

Data Type: unsigned16

Data Type Semantics: identifier

References: [RFC3261]

ElementId: TBD

Replaces Enterprise-Specific Element: 35566 / 412

A.2. duplicatePacketDeltaCount

Description: The number of uncorrupted and identical additional copies of each individual packet in the Flow arriving at the destination since the previous Data Record for this Flow (if any), as measured at the Observation Point. This is measured as the Type-P-one-way-packet-duplication metric defined in section 3 of [RFC5560].

Data Type: unsigned64

Data Type Semantics: deltaCounter

Units: packets

References: [RFC5560]

ElementId: TBD

A.3. ambientTemperature

Description: An ambient temperature observed by measurement equipment at an Observation Point, positioned such that it measures the temperature of the surroundings (i.e., not including any heat generated by the measuring or measured equipment), expressed in degrees Celsius.

Data Type: float

Units: degrees Celsius

Range: -273.15 - +inf

ElementId: TBD

Authors' Addresses

Brian Trammell
Swiss Federal Institute of Technology Zurich
Gloriastrasse 35
8092 Zurich
Switzerland

Phone: +41 44 632 70 13
Email: trammell@tik.ee.ethz.ch

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

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

Network Working Group
Internet Draft
Obsoletes: 5102
Category: Standards Track
Expires: January 14, 2013

B. Claise, Ed.
Cisco Systems, Inc.
B. Trammell, Ed.
ETH Zurich
July 13, 2012

Information Model for IP Flow Information eXport (IPFIX)
draft-ietf-ipfix-information-model-rfc5102bis-03.txt

Abstract

This document provides an overview of the information model for the IP Flow Information eXport (IPFIX) protocol, as defined in the IANA IPFIX Information Element Registry. It is used by the IPFIX Protocol for encoding measured traffic information and information related to the traffic Observation Point, the traffic Metering Process, and the Exporting Process. Although developed for the IPFIX Protocol, the model is defined in an open way that easily allows using it in other protocols, interfaces, and applications. This document obsoletes RFC 5102.

Status of This Memo

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

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

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

This Internet-Draft will expire on March 23, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Changes since RFC 5102	4
1.2. IPFIX Documents Overview	4
2. Properties of IPFIX Protocol Information Elements	5
2.1. Information Element Specification Template	5
2.2. Scope of Information Elements	7
2.3. Naming Conventions for Information Elements	8
3. Type Space	9
3.1. Abstract Data Types	9
3.1.1. unsigned8	9
3.1.2. unsigned16	9
3.1.3. unsigned32	9
3.1.4. unsigned64	9
3.1.5. signed8	9
3.1.6. signed16	10
3.1.7. signed32	10
3.1.8. signed64	10
3.1.9. float32	10
3.1.10. float64	10
3.1.11. boolean	10
3.1.12. macAddress	10
3.1.13. octetArray	10
3.1.14. string	10
3.1.15. dateTimeSeconds	11
3.1.16. dateTimeMilliseconds	11
3.1.17. dateTimeMicroseconds	11
3.1.18. dateTimeNanoseconds	11
3.1.19. ipv4Address	11
3.1.20. ipv6Address	11
3.2. Data Type Semantics	11
3.2.1. quantity	11
3.2.2. totalCounter	12
3.2.3. deltaCounter	12
3.2.4. identifier	12
3.2.5. flags	12
4. Information Element Identifiers	12
4.1. NetFlow version 9 compatible Information Element Identifiers	13

5.	Information Element Categories	15
5.1.	Identifiers	16
5.3.	Metering and Exporting Process Statistics	17
5.4.	IP Header Fields	17
5.5.	Transport Header Fields	18
5.6.	Sub-IP Header Fields	19
5.7.	Derived Packet Properties	19
5.9.	Flow Timestamps	20
5.10.	Per-Flow Counters	20
5.11.	Miscellaneous Flow Properties	21
5.12.	Padding	22
6.	Extending the Information Model	22
7.	IANA Considerations	23
7.1.	IPFIX Information Elements	23
7.2.	MPLS Label Type Identifier	23
7.3.	XML Namespace and Schema	23
8.	Security Considerations	24
9.	Acknowledgements	25
10.	References	25
10.1.	Normative References	25
10.2.	Informative References	25
	Authors' Addresses	29

OPEN ISSUES:

review the NetFlow V9-compatible Information Elements table in Section 4 to ensure that it includes V9 IEs recently made compatible/non-proprietary: 82, 83, 91, 98, 99. What about deltaFlowCount (3)? On second thought, consider removing these tables, they add nothing and ignore the last five years.

1. Introduction

The IP Flow Information eXport (IPFIX) protocol serves for transmitting information related to measured IP traffic over the Internet. The protocol specification in [RFC5101bis] defines how Information Elements are transmitted. For Information Elements, it specifies the encoding of a set of basic data types. However, the list of Information Elements that can be transmitted by the protocol, such as Flow attributes (source IP address, number of packets, etc.) and information about the Metering and Exporting Process (packet Observation Point, sampling rate, Flow timeout interval, etc.), is not specified in [RFC5101bis].

The canonical reference for IPFIX Information Elements the IANA IPFIX Information Element registry [IPFIX-IANA]; the initial values for

this registry were provided by [RFC5102].

This document complements the IPFIX protocol specification by providing an overview of the IPFIX information model and specifying data types for it. IPFIX-specific terminology used in this document is defined in Section 2 of [RFC5101bis]. As in [RFC5101bis], these IPFIX-specific terms have the first letter of a word capitalized when used in this document.

The use of the term 'information model' is not fully in line with the definition of this term in [RFC3444]. The IPFIX information model does not specify relationships between Information Elements, but also it does not specify a concrete encoding of Information Elements. Besides the encoding used by the IPFIX protocol, other encodings of IPFIX Information Elements can be applied, for example, XML-based encodings.

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. Changes since RFC 5102

This document obsoletes the Proposed Standard revision of the IPFIX Protocol Specification [RFC5102]. The following changes have been made to this document with respect to the previous document:

- EDITOR'S NOTE: not sure if we need to this information

- Errata ID: 1307 (technical)

- Errata ID: 1492 (technical)

- Errata ID: 1736 (technical)

- Errata ID: 2879 (editorial)

- Errata ID: 2944, which updates 1737 (technical)

- Errata ID: 2945, which updates 1738 (technical)

- Errata ID: 2946, which updates 1739 (technical)

- Updated the reference to RFC5101bis

- Clarified the time-related IEs

- Since this document is based on the IPFIX Draft Standard [RFC5101bis], all improvements have been taken into account. For example, the timestamps.- Instead of repeating every Information Elements from [RFC5102], a reference to the IPFIX IANA registry [IPFIX-IANA] is introduced. However the category in section 5 have been kept.- The appendix A and B have been removed- Introduced [IPFIX-IE-DOCTORS]

1.2. IPFIX Documents Overview

The IPFIX protocol 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 specifications [RFC5101bis] document specifies how IPFIX data records and templates are carried via a number of transport protocols from IPFIX Exporting Processes to IPFIX Collecting Processes.

Four IPFIX optimizations/extensions are currently specified: a bandwidth saving method for the IPFIX protocol in [RFC5473], an efficient method for exporting bidirectional flow in [RFC5103], a method for the definition and export of complex data structures in [RFC6313], and the specification of the Protocol for IPFIX Mediations [IPFIX-MED-PROTO] based on the IPFIX Mediation Framework [RFC6183].

IPFIX has a formal description of IPFIX Information Elements, their name, type and additional semantic information, as specified in this document, with the export of the Information Element types specified in [RFC5610].

[IPFIX-CONF] specifies a data model for configuring and monitoring IPFIX and PSAMP compliant devices using the NETCONF protocol, while the [RFC5815bis] specifies a MIB module for monitoring.

In terms of development, [RFC5153] provides guidelines for the implementation and use of the IPFIX protocol, while [RFC5471] provides guidelines for testing.

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.

2. Properties of IPFIX Protocol Information Elements

2.1. Information Element Specification Template

Information in messages of the IPFIX protocol is modeled in terms of Information Elements of the IPFIX information model. The IPFIX Information Elements mentioned in Section 5 are specified in [IPFIX-IANA]. For specifying these Information Elements, a template is used that is described below.

All Information Elements specified for the IPFIX protocol MUST have the following properties defined:

name - A unique and meaningful name for the Information Element.

`elementId` - A numeric identifier of the Information Element. If this identifier is used without an enterprise identifier (see [RFC5101bis] and `enterpriseId` below), then it is globally unique and the list of allowed values is administered by IANA. It is used for compact identification of an Information Element when encoding Templates in the protocol.

`description` - The semantics of this Information Element. Describes how this Information Element is derived from the Flow or other information available to the observer. Information Elements of `dataType` string or `octetArray` which have a length constraints (fixed length, minimum and/or maximum length) MUST note these constraints in their description.

`dataType` - One of the types listed in Section 3.1 of this document or registered in the IANA IPFIX Information Element Data Types registry. The type space for attributes is constrained to facilitate implementation. The existing type space does however encompass most basic types used in modern programming languages, as well as some derived types (such as `ipv4Address`) that are common to this domain and useful to distinguish.

`status` - The status of the specification of this Information Element. Allowed values are 'current', 'deprecated', and 'obsolete'. All newly-defined Information Elements have 'current' status. The process for moving Information Elements to the 'deprecated' or 'obsolete' status is defined in Section 5.2 of [IPFIX-IE-DOCTORS].

Enterprise-specific Information Elements MUST have the following property defined:

`enterpriseId` - Enterprises may wish to define Information Elements without registering them with IANA, for example, for enterprise-internal purposes. For such Information Elements, the Information Element identifier described above is not sufficient when the Information Element is used outside the enterprise. If specifications of enterprise-specific Information Elements are made public and/or if enterprise-specific identifiers are used by the IPFIX protocol outside the enterprise, then the enterprise-specific identifier MUST be made globally unique by combining it with an enterprise identifier. Valid values for the `enterpriseId` are defined by IANA as Structure of Management Information (SMI) network management private enterprise codes. They are defined at <http://www.iana.org/assignments/enterprise-numbers>.

All Information Elements specified for the IPFIX protocol either in this document or by any future extension MAY have the following

properties defined:

dataTypeSemantics - The integral types may be qualified by additional semantic details. Valid values for the data type semantics are specified in Section 3.2 of this document or in a future extension of the information model.

units - If the Information Element is a measure of some kind, the units identify what the measure is.

range - Some Information Elements may only be able to take on a restricted set of values that can be expressed as a range (e.g., 0 through 511 inclusive). If this is the case, the valid inclusive range should be specified.

reference - Identifies additional specifications that more precisely define this item or provide additional context for its use.

Information Elements replacing an enterprise-specific Information Element used for experimental or pre-standardization purposes SHOULD define the following property, as outlined in Section 4.9 of [IPFIX-IE-DOCTORS].

enterpriseSpecificReference - The `enterpriseId` and `elementId` of the enterprise-specific Information Element(s) replaced by this Information Element.

The following two Information Element properties are defined to allow the management of an Information Element registry with Information Element definitions that may be updated over time, per the process defined in Section 5.2 of [IPFIX-IE-DOCTORS].

revision - The revision number of an Information Element, starting at 0 for Information Elements at time of definition, and incremented by one for each revision.

date - The date of the entry of this revision of the Information Element into the registry.

For Information Elements of the string or `octetArray` data types which have size limits (minimum and/or maximum size, or fixed length), the limits MUST be defined within the description of the Information Element.

2.2. Scope of Information Elements

By default, most Information Elements have a scope specified in their definitions.

- o The Information Elements listed in Sections 5.2 and 5.3, and similar Information Elements in [IPFIX-IANA], have a default of "a specific Metering Process" or of "a specific Exporting Process", respectively.
- o The Information Elements listed in Sections 5.4-5.11, and similar Information Elements in [IPFIX-IANA], have a scope of "a specific Flow".

Within Data Records defined by Option Templates, the IPFIX protocol allows further limiting of the Information Element scope. The new scope is specified by one or more scope fields and defined as the combination of all specified scope values; see Section 3.4.2.1 on IPFIX scopes in [RFC5101bis].

2.3. Naming Conventions for Information Elements

The following naming conventions were used for naming Information Elements in this document. It is recommended that extensions of the model use the same conventions.

- o Names of Information Elements SHOULD be descriptive.
- o Names of Information Elements MUST be unique within the IANA registry. Enterprise-specific Information Elements SHOULD be prefixed with a vendor name.
- o Names of Information Elements MUST start with non-capitalized letters.
- o Composed names MUST use capital letters for the first letter of each component (except for the first one). All other letters are non-capitalized, even for acronyms. Exceptions are made for acronyms containing non-capitalized letters, such as 'IPv4' and 'IPv6'. Examples are sourceMacAddress and destinationIPv4Address.
- o Middleboxes [RFC3234] may change Flow properties, such as the Differentiated Service Code Point (DSCP) value or the source IP address. If an IPFIX Observation Point is located in the path of a Flow before one or more middleboxes that potentially modify packets of the Flow, then it may be desirable to also report Flow properties after the modification performed by the middleboxes. An example is an Observation Point before a packet marker changing a packet's IPv4 Type of Service (TOS) field that is encoded in Information Element ipClassOfService. Then the value observed and reported by Information Element ipClassOfService is valid at the Observation Point, but not after the packet passed the packet marker. For reporting the change value of the TOS field, the

IPFIX information model uses Information Elements that have a name prefix "post", for example, "postIpClassOfService". Information Elements with prefix "post" report on Flow properties that are not necessarily observed at the Observation Point, but which are obtained within the Flow's Observation Domain by other means considered to be sufficiently reliable, for example, by analyzing the packet marker's marking tables.

3. Type Space

This section describes the abstract data types that can be used for the specification of IPFIX Information Elements in Section 4. Section 3.1 describes the set of abstract data types.

Abstract data types unsigned8, unsigned16, unsigned32, unsigned64, signed8, signed16, signed32, and signed64 are integral data types. As described in Section 3.2, their data type semantics can be further specified, for example, by 'totalCounter', 'deltaCounter', 'identifier', or 'flags'.

3.1. Abstract Data Types

This section describes the set of valid abstract data types of the IPFIX information model. Note that further abstract data types may be specified by future extensions of the IPFIX information model.

3.1.1. unsigned8

The type "unsigned8" represents a non-negative integer value in the range of 0 to 255.

3.1.2. unsigned16

The type "unsigned16" represents a non-negative integer value in the range of 0 to 65535.

3.1.3. unsigned32

The type "unsigned32" represents a non-negative integer value in the range of 0 to 4294967295.

3.1.4. unsigned64

The type "unsigned64" represents a non-negative integer value in the range of 0 to 18446744073709551615.

3.1.5. signed8

The type "signed8" represents an integer value in the range of -128 to 127.

3.1.6. signed16

The type "signed16" represents an integer value in the range of -32768 to 32767.

3.1.7. signed32

The type "signed32" represents an integer value in the range of -2147483648 to 2147483647.

3.1.8. signed64

The type "signed64" represents an integer value in the range of -9223372036854775808 to 9223372036854775807.

3.1.9. float32

The type "float32" corresponds to an IEEE single-precision 32-bit floating point type as defined in [IEEE.754.1985].

3.1.10. float64

The type "float64" corresponds to an IEEE double-precision 64-bit floating point type as defined in [IEEE.754.1985].

3.1.11. boolean

The type "boolean" represents a binary value. The only allowed values are "true" and "false".

3.1.12. macAddress

The type "macAddress" represents a string of 6 octets.

3.1.13. octetArray

The type "octetArray" represents a finite-length string of octets.

3.1.14. string

The type "string" represents a finite-length string of valid characters from the Unicode character encoding set [ISO.10646-1.1993]. Unicode allows for ASCII [ISO.646.1991] and many other international character sets to be used.

3.1.15. `dateTimeSeconds`

The data type `dateTimeSeconds` is an unsigned 32-bit integer representing the number of seconds since the UNIX epoch, 1 January 1970 at 00:00 UTC, as defined in [POSIX.1].

3.1.16. `dateTimeMilliseconds`

The data type `dateTimeMilliseconds` is an unsigned 64-bit integer containing the number of milliseconds since the UNIX epoch, 1 January 1970 at 00:00 UTC, as defined in [POSIX.1].

3.1.17. `dateTimeMicroseconds`

The type "`dateTimeMicroseconds`" represents a time value with microsecond precision according to the NTP Timestamp format as defined in section 6 of [RFC5905].

3.1.18. `dateTimeNanoseconds`

The type "`dateTimeNanoseconds`" represents a time value with nanosecond precision according to the NTP Timestamp format as defined in section 6 of [RFC5905].

3.1.19. `ipv4Address`

The type "`ipv4Address`" represents a value of an IPv4 address.

3.1.20. `ipv6Address`

The type "`ipv6Address`" represents a value of an IPv6 address.

3.2. Data Type Semantics

This section describes the set of valid data type semantics of the IPFIX information model. A registry of data type semantics is established in [RFC5610]; the restrictions specified in section 3.10 of that document are followed here. Note that further data type semantics may be specified by future extensions of the IPFIX information model. These semantics apply only to numeric types, as noted in the description of each semantic below.

3.2.1. `quantity`

A numeric (integral or floating point) value representing a measured value pertaining to the record. This is distinguished from counters that represent an ongoing measured value whose "odometer" reading is captured as part of a given record. This is the default semantic type

of all numeric data types.

3.2.2. totalCounter

An numeric value reporting the value of a counter. Counters are unsigned and wrap back to zero after reaching the limit of the type. For example, an unsigned64 with counter semantics will continue to increment until reaching the value of $2^{64} - 1$. At this point, the next increment will wrap its value to zero and continue counting from zero. The semantics of a total counter is similar to the semantics of counters used in SNMP, such as Counter32 defined in [RFC2578]. The only difference between total counters and counters used in SNMP is that the total counters have an initial value of 0. A total counter counts independently of the export of its value.

3.2.3. deltaCounter

An numeric value reporting the value of a counter. Counters are unsigned and wrap back to zero after reaching the limit of the type. For example, an unsigned64 with counter semantics will continue to increment until reaching the value of $2^{64} - 1$. At this point, the next increment will wrap its value to zero and continue counting from zero. The semantics of a delta counter is similar to the semantics of counters used in SNMP, such as Counter32 defined in RFC 2578 [RFC2578]. The only difference between delta counters and counters used in SNMP is that the delta counters have an initial value of 0. A delta counter is reset to 0 each time its value is exported.

3.2.4. identifier

An integral value that serves as an identifier. Specifically, mathematical operations on two identifiers (aside from the equality operation) are meaningless. For example, Autonomous System ID 1 * Autonomous System ID 2 is meaningless. Identifiers MUST be one of the signed or unsigned data types.

3.2.5. flags

An integral value that represents a set of bit fields. Logical operations are appropriate on such values, but not other mathematical operations. Flags MUST always be of an unsigned data type.

4. Information Element Identifiers

All Information Elements defined in the IANA IPFIX Information Element registry [IPFIX-IANA] have their identifiers assigned by IANA.

The value of these identifiers is in the range of 1-32767. Within this range, Information Element identifier values in the sub-range of 1-127 are compatible with field types used by NetFlow version 9 [RFC3954]; Information Element identifiers in this range MUST NOT be assigned unless the Information Element is compatible with the NetFlow version 9 protocol. Such Information Elements may ONLY be requested by a NetFlow v9 expert, to be designated by the IESG.

In general, IANA will add newly registered Information Elements to the registry, assigning the lowest available Information Element identifier in the range 128-32767.

Enterprise-specific Information Element identifiers have the same range of 1-32767, but they are coupled with an additional enterprise identifier. For enterprise-specific Information Elements, Information Element identifier 0 is also reserved. Enterprise-specific Information Element identifiers can be chosen by an enterprise arbitrarily within the range of 1-32767. The same identifier may be assigned by other enterprises for different purposes; these Information Elements are distinct because the Information Element identifier is coupled with an enterprise identifier.

Enterprise identifiers MUST be registered as SMI network management private enterprise code numbers with IANA. The registry can be found at <http://www.iana.org/assignments/enterprise-numbers>.

4.1. NetFlow version 9 compatible Information Element Identifiers

The following list gives an overview of the Information Element identifiers that are compatible with field types used by NetFlow version 9 [RFC3954].

ID	Name	ID	Name
1	octetDeltaCount	43	RESERVED
2	packetDeltaCount	44	sourceIPv4Prefix
3	RESERVED	45	destinationIPv4Prefix
4	protocolIdentifier	46	mplsTopLabelType
5	ipClassOfService	47	mplsTopLabelIPv4Address
6	tcpControlBits	48-51	RESERVED
7	sourceTransportPort	52	minimumTTL
8	sourceIPv4Address	53	maximumTTL
9	sourceIPv4PrefixLength	54	fragmentIdentification
10	ingressInterface	55	postIpClassOfService
11	destinationTransportPort	56	sourceMacAddress
12	destinationIPv4Address	57	postDestinationMacAddress
13	destinationIPv4PrefixLength	58	vlanId
14	egressInterface	59	postVlanId
15	ipNextHopIPv4Address	60	ipVersion
16	bgpSourceAsNumber	61	flowDirection
17	bgpDestinationAsNumber	62	ipNextHopIPv6Address
18	bgpNextHopIPv4Address	63	bgpNextHopIPv6Address
19	postMcastPacketDeltaCount	64	ipv6ExtensionHeaders
20	postMcastOctetDeltaCount	65-69	RESERVED
21	flowEndSysUpTime	70	mplsTopLabelStackSection
22	flowStartSysUpTime	71	mplsLabelStackSection2
23	postOctetDeltaCount	72	mplsLabelStackSection3
24	postPacketDeltaCount	73	mplsLabelStackSection4
25	minimumIpTotalLength	74	mplsLabelStackSection5
26	maximumIpTotalLength	75	mplsLabelStackSection6
27	sourceIPv6Address	76	mplsLabelStackSection7
28	destinationIPv6Address	77	mplsLabelStackSection8
29	sourceIPv6PrefixLength	78	mplsLabelStackSection9
30	destinationIPv6PrefixLength	79	mplsLabelStackSection10
31	flowLabelIPv6	80	destinationMacAddress
32	icmpTypeCodeIPv4	81	postSourceMacAddress
33	igmpType	82-84	RESERVED
34	RESERVED	85	octetTotalCount
35	RESERVED	86	packetTotalCount
36	flowActiveTimeout	87	RESERVED
37	flowIdleTimeout	88	fragmentOffset
38	RESERVED	89	RESERVED
39	RESERVED	90	mplsVpnRouteDistinguisher
40	exportedOctetTotalCount	91-127	RESERVED
41	exportedMessageTotalCount		
42	exportedFlowRecordTotalCount		

5. Information Element Categories

This section describes the Information Element category for the IPFIX information model at the time that [RFC5102] was published. Since this category field is not part of the IANA process for assigning new Information Element (even though it has been reused, for example, in [RFC5103]), the newest Information Elements in IANA [IPFIX-IANA] don't have this classification. The elements are grouped into 12 groups according to their semantics and their applicability:

1. Identifiers
2. Metering and Exporting Process Configuration
3. Metering and Exporting Process Statistics
4. IP Header Fields
5. Transport Header Fields
6. Sub-IP Header Fields
7. Derived Packet Properties
8. Min/Max Flow Properties
9. Flow Timestamps
10. Per-Flow Counters
11. Miscellaneous Flow Properties
12. Padding

The Information Elements that are derived from fields of packets or from packet treatment, such as the Information Elements in groups 4-7, can typically serve as Flow Keys used for mapping packets to Flows.

If they do not serve as Flow Keys, their value may change from packet to packet within a single Flow. For Information Elements with values that are derived from fields of packets or from packet treatment and for which the value may change from packet to packet within a single Flow, the IPFIX information model defines that their value is determined by the first packet observed for the corresponding Flow, unless the description of the Information Element explicitly specifies a different semantics. This simple rule allows writing all Information Elements related to header fields once when the first packet of the Flow is observed. For further observed packets of the same Flow, only Flow properties that depend on more than one packet, such as the Information Elements in groups 8-11, need to be updated.

Information Elements with a name having the "post" prefix, for example, "postIpClassOfService", do not report properties that were actually observed at the Observation Point, but retrieved by other means within the Observation Domain. These Information Elements can be used if there are middlebox functions within the Observation Domain changing Flow properties after packets passed the Observation Point.

5.1. Identifiers

Information Elements grouped in the table below are identifying components of the IPFIX architecture, of an IPFIX Device, or of the IPFIX protocol. All of them have an integral abstract data type and data type semantics "identifier" as described in Section 3.2.4.

Typically, some of them are used for limiting scopes of other Information Elements. However, other Information Elements MAY be used for limiting scopes. Note also that all Information Elements listed below MAY be used for other purposes than limiting scopes.

ID	Name	ID	Name
141	lineCardId	148	flowId
142	portId	145	templateId
10	ingressInterface	149	observationDomainId
14	egressInterface	138	observationPointId
143	meteringProcessId	137	commonPropertiesId
144	exportingProcessId		

See [IPFIX-IANA] for the definitions of these Information Elements.

5.2. Metering and Exporting Process Configuration

Information Elements in this section describe the configuration of the Metering Process or the Exporting Process. The set of these Information Elements is listed in the table below.

ID	Name	ID	Name
130	exporterIPv4Address	213	exportInterface
131	exporterIPv6Address	214	exportProtocolVersion
217	exporterTransportPort	215	exportTransportProtocol
211	collectorIPv4Address	216	collectorTransportPort
212	collectorIPv6Address	173	flowKeyIndicator

See [IPFIX-IANA] for the definitions of these Information Elements.

5.3. Metering and Exporting Process Statistics

Information Elements in this section describe statistics of the Metering Process and/or the Exporting Process. The set of these Information Elements is listed in the table below.

ID	Name	ID	Name
41	exportedMessageTotalCount	165	ignoredOctetTotalCount
40	exportedOctetTotalCount	166	notSentFlowTotalCount
42	exportedFlowRecordTotalCount	167	notSentPacketTotalCount
163	observedFlowTotalCount	168	notSentOctetTotalCount
164	ignoredPacketTotalCount		

See [IPFIX-IANA] for the definitions of these Information Elements.

5.4. IP Header Fields

Information Elements in this section indicate values of IP header fields or are derived from IP header field values in combination with further information.

ID	Name	ID	Name
60	ipVersion	193	nextHeaderIPv6
8	sourceIPv4Address	195	ipDiffServCodePoint
27	sourceIPv6Address	196	ipPrecedence
9	sourceIPv4PrefixLength	5	ipClassOfService
29	sourceIPv6PrefixLength	55	postIpClassOfService
44	sourceIPv4Prefix	31	flowLabelIPv6
170	sourceIPv6Prefix	206	isMulticast
12	destinationIPv4Address	54	fragmentIdentification
28	destinationIPv6Address	88	fragmentOffset
13	destinationIPv4PrefixLength	197	fragmentFlags
30	destinationIPv6PrefixLength	189	ipHeaderLength
45	destinationIPv4Prefix	207	ipv4IHL
169	destinationIPv6Prefix	190	totalLengthIPv4
192	ipTTL	224	ipTotalLength
4	protocolIdentifier	191	payloadLengthIPv6

See [IPFIX-IANA] for the definitions of these Information Elements.

5.5. Transport Header Fields

The set of Information Elements related to transport header fields and length includes the Information Elements listed in the table below.

ID	Name	ID	Name
7	sourceTransportPort	238	tcpWindowScale
11	destinationTransportPort	187	tcpUrgentPointer
180	udpSourcePort	188	tcpHeaderLength
181	udpDestinationPort	32	icmpTypeCodeIPv4
205	udpMessageLength	176	icmpTypeIPv4
182	tcpSourcePort	177	icmpCodeIPv4
183	tcpDestinationPort	139	icmpTypeCodeIPv6
184	tcpSequenceNumber	178	icmpTypeIPv6
185	tcpAcknowledgementNumber	179	icmpCodeIPv6
186	tcpWindowSize	33	igmpType

See [IPFIX-IANA] for the definitions of these Information Elements.

5.6. Sub-IP Header Fields

The set of Information Elements related to Sub-IP header fields includes the Information Elements listed in the table below.

ID	Name	ID	Name
56	sourceMacAddress	201	mplsLabelStackLength
81	postSourceMacAddress	194	mplsPayloadLength
58	vlanId	70	mplsTopLabelStackSection
59	postVlanId	71	mplsLabelStackSection2
80	destinationMacAddress	72	mplsLabelStackSection3
57	postDestinationMacAddress	73	mplsLabelStackSection4
146	wlanChannelId	74	mplsLabelStackSection5
147	wlanSSID	75	mplsLabelStackSection6
200	mplsTopLabelTTL	76	mplsLabelStackSection7
203	mplsTopLabelExp	77	mplsLabelStackSection8
237	postMplsTopLabelExp	78	mplsLabelStackSection9
202	mplsLabelStackDepth	79	mplsLabelStackSection10

See [IPFIX-IANA] for the definitions of these Information Elements.

5.7. Derived Packet Properties

The set of Information Elements derived from packet properties (for example, values of header fields) includes the Information Elements listed in the table below.

ID	Name	ID	Name
204	ipPayloadLength	18	bgpNextHopIPv4Address
15	ipNextHopIPv4Address	63	bgpNextHopIPv6Address
62	ipNextHopIPv6Address	46	mplsTopLabelType
16	bgpSourceAsNumber	47	mplsTopLabelIPv4Address
17	bgpDestinationAsNumber	140	mplsTopLabelIPv6Address
128	bgpNextAdjacentAsNumber	90	mplsVpnRouteDistinguisher
129	bgpPrevAdjacentAsNumber		

See [IPFIX-IANA] for the definitions of these Information Elements.

5.9. Flow Timestamps

Information Elements in this section are timestamps of events.

Timestamps `flowStartSeconds`, `flowEndSeconds`, `flowStartMilliseconds`, `flowEndMilliseconds`, `flowStartMicroseconds`, `flowEndMicroseconds`, `flowStartNanoseconds`, `flowEndNanoseconds`, and `systemInitTimeMilliseconds` are absolute and have a well-defined fixed time base, such as, for example, the number of seconds since 0000 UTC Jan 1st 1970.

Timestamps `flowStartDeltaMicroseconds` and `flowEndDeltaMicroseconds` are relative timestamps only valid within the scope of a single IPFIX Message. They contain the negative time offsets relative to the export time specified in the IPFIX Message Header. The maximum time offset that can be encoded by these delta counters is 1 hour, 11 minutes, and 34.967295 seconds.

Timestamps `flowStartSysUpTime` and `flowEndSysUpTime` are relative timestamps indicating the time relative to the last (re-)initialization of the IPFIX Device. For reporting the time of the last (re-)initialization, `systemInitTimeMilliseconds` can be reported, for example, in Data Records defined by Option Templates.

ID	Name	ID	Name
150	<code>flowStartSeconds</code>	156	<code>flowStartNanoseconds</code>
151	<code>flowEndSeconds</code>	157	<code>flowEndNanoseconds</code>
152	<code>flowStartMilliseconds</code>	158	<code>flowStartDeltaMicroseconds</code>
153	<code>flowEndMilliseconds</code>	159	<code>flowEndDeltaMicroseconds</code>
154	<code>flowStartMicroseconds</code>	160	<code>systemInitTimeMilliseconds</code>
155	<code>flowEndMicroseconds</code>	22	<code>flowStartSysUpTime</code>
		21	<code>flowEndSysUpTime</code>

See [IPFIX-IANA] for the definitions of these Information Elements.

5.10. Per-Flow Counters

Information Elements in this section are counters all having integer values. Their values may change for every report they are used in. They cannot serve as part of a Flow Key used for mapping packets to Flows. However, potentially they can be used for selecting exported Flows, for example, by only exporting Flows with more than a threshold number of observed octets.

There are running counters and delta counters. Delta counters are reset to zero each time their values are exported. Running counters continue counting independently of the Exporting Process.

There are per-Flow counters and counters related to the Metering Process and/or the Exporting Process. Per-Flow counters are Flow properties that potentially change each time a packet belonging to the Flow is observed. The set of per-Flow counters includes the Information Elements listed in the table below. Counters related to the Metering Process and/or the Exporting Process are described in Section 5.3.

ID	Name	ID	Name
1	octetDeltaCount	134	droppedOctetTotalCount
23	postOctetDeltaCount	135	droppedPacketTotalCount
198	octetDeltaSumOfSquares	19	postMCastPacketDeltaCount
85	octetTotalCount	20	postMCastOctetDeltaCount
171	postOctetTotalCount	174	postMCastPacketTotalCount
199	octetTotalSumOfSquares	175	postMCastOctetTotalCount
2	packetDeltaCount	218	tcpSynTotalCount
24	postPacketDeltaCount	219	tcpFinTotalCount
86	packetTotalCount	220	tcpRstTotalCount
172	postPacketTotalCount	221	tcpPshTotalCount
132	droppedOctetDeltaCount	222	tcpAckTotalCount
133	droppedPacketDeltaCount	223	tcpUrgTotalCount

See [IPFIX-IANA] for the definitions of these Information Elements.

5.11. Miscellaneous Flow Properties

Information Elements in this section describe properties of Flows that are related to Flow start, Flow duration, and Flow termination, but they are not timestamps as the Information Elements in Section 5.9 are.

ID	Name	ID	Name
36	flowActiveTimeout	161	flowDurationMilliseconds
37	flowIdleTimeout	162	flowDurationMicroseconds
136	flowEndReason	61	flowDirection

See [IPFIX-IANA] for the definitions of these Information Elements.

5.12. Padding

This section contains a single Information Element that can be used for padding of Flow Records.

IPFIX implementations may wish to align Information Elements within Data Records or to align entire Data Records to 4-octet or 8-octet boundaries. This can be achieved by including one or more paddingOctets Information Elements in a Data Record.

ID	Name	ID	Name
210	paddingOctets		

See [IPFIX-IANA] for the definitions of these Information Elements.

6. Extending the Information Model

A key requirement for IPFIX is to allow for extension of the Information Model maintained by IANA. The process for extending the Information Model is defined in [IPFIX-IE-DOCTORS], which also provides guidelines for authors and reviewers of new Information Element definitions.

For new Information Elements, the type space defined in Section 3 can be used. If required, new abstract data types can be added to the subregistry defined in [RFC5610]. New abstract data types MUST be defined in IETF Standards Track documents.

Enterprises may wish to define Information Elements without registering them with IANA. IPFIX explicitly supports enterprise-specific Information Elements. Enterprise-specific Information Elements are described in Sections 2.1 and 4; guidelines for using them appear in [IPFIX-IE-DOCTORS].

7. IANA Considerations

7.1. IPFIX Information Elements

This document refers to Information Elements, for which the Internet Assigned Numbers Authority (IANA) has created the IPFIX Information Element Registry [IPFIX-IANA]. The columns of this registry must at minimum be able to store the information defined in the template in Section 2.1., additional columns defined in [IPFIX-IE-DOCTORS]; it may contain other information as necessary for the management of the registry.

New assignments for IPFIX Information Elements will be administered by IANA through Expert Review [RFC5226], i.e., review by one of a group of experts designated by an IETF Area Director. Further considerations for this review are specified in [IPFIX-IE-DOCTORS].

[NOTE to IANA: please update the Reference for the IPFIX Information Element Registry to refer to this document, as well as to [IPFIX-IE-DOCTORS].]

7.2. MPLS Label Type Identifier

Information Element #46, named `mplsTopLabelType`, carries MPLS label types. Values for 5 different types have initially been defined. For ensuring extensibility of this information, IANA has created a new subregistry for MPLS label types and filled it with the initial list from the description Information Element #46, `mplsTopLabelType`.

New assignments for MPLS label types will be administered by IANA through Expert Review [RFC5226], i.e., review by one of a group of experts designated by an IETF Area Director. The group of experts must double check the label type definitions with already defined label types for completeness, accuracy, and redundancy. The specification of new MPLS label types MUST be published using a well-established and persistent publication medium.

[NOTE to IANA: please update the Reference for the IPFIX MPLS Label Type subregistry to refer to this document.]

7.3. XML Namespace and Schema

[IPFIX-XML-SCHEMA] defines an XML schema for IPFIX Information Element definitions. All Information Elements specified in [IPFIX-IANA] are defined by this schema. This schema may also be used for specifying further Information Elements in future extensions of the IPFIX information model in a machine-readable way.

[IPFIX-XML-SCHEMA] uses URNs to describe an XML namespace and an XML schema for IPFIX Information Elements conforming to a registry mechanism described in [RFC3688]. Two URI assignments have been made.

1. Registration for the IPFIX information model namespace
 - * URI: urn:ietf:params:xml:ns:ipfix-info
 - * Registrant Contact: IETF IPFIX Working Group <ipfix@ietf.org>, as designated by the IESG <iesg@ietf.org>.
 - * XML: None. Namespace URIs do not represent an XML.
2. Registration for the IPFIX information model schema
 - * URI: urn:ietf:params:xml:schema:ipfix-info
 - * Registrant Contact: IETF IPFIX Working Group <ipfix@ietf.org>, as designated by the IESG <iesg@ietf.org>.

Using a machine-readable syntax for the information model enables the creation of IPFIX-aware tools that can automatically adapt to extensions to the information model, by simply reading updated information model specifications.

The wide availability of XML-aware tools and libraries for client devices is a primary consideration for this choice. In particular, libraries for parsing XML documents are readily available. Also, mechanisms such as the Extensible Stylesheet Language (XSL) allow for transforming a source XML document into other documents. This document was authored in XML and transformed according to [RFC2629].

It should be noted that the use of XML in Exporters, Collectors, or other tools is not mandatory for the deployment of IPFIX. In particular, Exporting Processes do not produce or consume XML as part of their operation. It is expected that IPFIX Collectors MAY take advantage of the machine readability of the information model vs. hard coding their behavior or inventing proprietary means for accommodating extensions.

8. Security Considerations

The IPFIX information model itself does not directly introduce security issues. Rather, it defines a set of attributes that may for privacy or business issues be considered sensitive information.

For example, exporting values of header fields may make attacks possible for the receiver of this information, which would otherwise only be possible for direct observers of the reported Flows along the data path.

The underlying protocol used to exchange the information described here must therefore apply appropriate procedures to guarantee the integrity and confidentiality of the exported information. Such protocols are

defined in separate documents, specifically the IPFIX protocol document [RFC5101bis].

This document does not specify any Information Element carrying keying material. If future extensions will do so, then appropriate precautions need to be taken for properly protecting such sensitive information.

9. Acknowledgements

The editors would like to thanks the authors of the RFC5102 [RFC5102], as this document is directly based upon this original RFC: Juergen Quittek, Stewart Bryant, Paul Aitken, and Jeff Meyer.

10. References

10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC5905] Mills, D., Delaware, U., Martin, J., Burbank, J. and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010
- [RFC5101bis] Claise, B., and B. Trammell, Editors, "Specification of the IP Flow Information eXport (IPFIX) Protocol for the Exchange of IP Traffic Flow Information", draft-ietf-ipfix-protocol-rfc5101bis-00, Work in Progress, November 2011.
- [IPFIX-IE-DOCTORS] Trammell, B., and B. Claise, "Guidelines for Authors and Reviewers of IPFIX Information Elements", draft-ietf-ipfix-ie-doctors-00, Work in Progress, November 2011.

10.2. Informative References

- [IEEE.754.1985] Institute of Electrical and Electronics Engineers, "Standard for Binary Floating-Point Arithmetic", IEEE Standard 754, August 1985.

- [ISO.10646-1.1993]
International Organization for Standardization,
"Information Technology - Universal Multiple-octet coded
Character Set (UCS) - Part 1: Architecture and Basic
Multilingual Plane", ISO Standard 10646-1, May 1993.
- [ISO.646.1991]
International Organization for Standardization,
"Information technology - ISO 7-bit coded character set
for information interchange", ISO Standard 646, 1991.
- [POSIX.1] IEEE 1003.1-2008 - IEEE Standard for Information
Technology - Portable Operating System Interface, IEEE,
2008.
- [RFC2578] McCloghrie, K., Perkins, D., and J. Schoenwaelder,
"Structure of Management Information Version 2 (SMIv2)",
STD 58, RFC 2578, April 1999.
- [RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629,
June 1999.
- [RFC3234] Carpenter, B. and S. Brim, "Middleboxes: Taxonomy and
Issues", RFC 3234, February 2002.
- [RFC3444] Pras, A. and J. Schoenwaelder, "On the Difference between
Information Models and Data Models", RFC 3444, January
2003.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688,
January 2004.
- [RFC3917] Quittek, J., Zseby, T., Claise, B., and S. Zander,
"Requirements for IP Flow Information Export (IPFIX)", RFC
3917, October 2004.
- [RFC3954] Claise, B., Ed., "Cisco Systems NetFlow Services Export
Version 9", RFC 3954, October 2004.
- [RFC5102] Trammell, B., and E. Boschi, "Bidirectional Flow Export
Using IP Flow Information Export (IPFIX)", RFC 5103,
January 2008.
- [RFC5103] Quittek, J., Bryant, S., Claise, B., Aitken, P., and J.
Meyer, "Information Model for IP Flow Information Export",
RFC 5102, January 2008.

- [RFC5153] Boschi, E., Mark, L., Quittek J., and P. Aitken, "IP Flow Information Export (IPFIX) Implementation Guidelines", RFC5153, April 2008.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.
- [RFC5470] Sadasivan, G., Brownlee, N., Claise, B., and J. Quittek, "Architecture for IP Flow Information Export", RFC5470, March 2009.
- [RFC5471] Schmoll, C., Aitken, P., and B. Claise, "Guidelines for IP Flow Information Export (IPFIX) Testing", RFC5471, March 2009.
- [RFC5472] Zseby, T., Boschi, E., Brownlee, N., and B. Claise, "IP Flow Information Export (IPFIX) Applicability", RFC5472, March 2009.
- [RFC5473] Boschi, E., Mark, L., and B. Claise, "Reducing Redundancy in IP Flow Information Export (IPFIX) and Packet Sampling (PSAMP) Reports", RFC5473, March 2009.
- [RFC5610] Boschi, E., Trammell, B., Mark, L., and T. Zseby, "Exporting Type Information for IP Flow Information Export (IPFIX) Information Elements", July 2009.
- [RFC6313] Claise, B., Dhandapani, G., Aitken, P, and S. Yates, "Export of Structured Data in IP Flow Information Export (IPFIX)", RFC6313, July 2011.
- [RFC6183] Kobayashi, A., Claise, B., Muenz, G, and K. Ishibashi, "IP Flow Information Export (IPFIX) Mediation: Framework", RFC6183, April 2011.
- [IPFIX-CONF]
Muenz, G., Claise, B., and P. Aitken, "Configuration Data Model for IPFIX and PSAMP", draft-ietf-ipfix-configuration-model-10, Work in Progress, July 2011.
- [IPFIX-MED-PROTO]
Claise, B., Kobayashi, A., and B. Trammell, "Specification of the Protocol for IPFIX Mediations", draft-ietf-ipfix-mediation-protocol-00, Work in Progress, December 2011.
- [RFC5815bis]
Dietz, T., Kobayashi, A., Claise, B., and G. Muenz,

"Definitions of Managed Objects for IP Flow Information Export", draft-ietf-ipfix-rfc5815bis-01.txt, Work in Progress, January 2012.

[IPFIX-IANA]

<http://www.iana.org/assignments/ipfix/ipfix.xml>

[IPFIX-XML-SCHEMA]

<http://www.iana.org/assignments/xml-registry/schema/ipfix.xsd>

Authors' Addresses

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

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

Brian Trammell
Swiss Federal Institute of Technology Zurich
Gloriastrasse 35
8092 Zurich
Switzerland

Phone: +41 44 632 70 13
EMail: trammell@tik.ee.ethz.ch

IPFIX Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 7, 2013

B. Claise
Cisco Systems, Inc.
A. Kobayashi
NTT
B. Trammell
ETH Zurich
July 6, 2012

Operation of the IP Flow Information Export (IPFIX) Protocol on IPFIX
Mediators
draft-ietf-ipfix-mediation-protocol-02.txt

Abstract

This document specifies the the operation of the IP Flow Information Export (IPFIX) protocol specific to IPFIX Mediators, including Template and Observation Point management, timing considerations, and other Mediator-specific concerns.

Status of this Memo

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

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

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

This Internet-Draft will expire on January 7, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. IPFIX Documents Overview	3
1.2. IPFIX Mediator Documents Overview	4
1.3. Relationship with IPFIX and PSAMP	5
2. Terminology	5
3. Handling IPFIX Message Headers	8
4. Template Management	10
4.1. Passing Unmodified Templates through a Mediator	10
4.2. Creating New Templates at a Mediator	14
4.3. Information Element Ordering within Templates	15
4.4. Handling Unknown Information Elements	15
5. Preserving Original Observation Point Information	15
5.1. originalExporterIPv4Address Information Element	17
5.2. originalExporterIPv6Address Information Element	17
6. Managing Observation Domain IDs	18
6.1. originalObservationDomainId Information Element	18
7. Timing Considerations	19
8. Transport Considerations	20
9. Collecting Process Considerations	20
10. Specific Reporting Requirements	21
10.1. Intermediate Process Reliability Statistics Template	21
10.2. Flow Key Options Template	22
10.3. intermediateProcessId Information Element	23
10.4. ignoredRecordTotalCount Information Element	23
11. Configuration Management	23
12. Security Considerations	24
13. IANA Considerations	24
14. Acknowledgments	24
15. References	25
15.1. Normative References	25
15.2. Informative References	26
Authors' Addresses	27

1. Introduction

The IPFIX architectural components in [RFC5470] consist of IPFIX Devices and IPFIX Collectors communicating using the IPFIX protocol [I-D.ietf-ipfix-protocol-rfc5101bis], which specifies how to export IP Flow information. This protocol is designed to export information about IP traffic Flows and related measurement data, where a Flow is defined by a set of key attributes (e.g. source and destination IP address, source and destination port, etc.).

However, thanks to its Template mechanism, the IPFIX protocol can export any type of information, as long as the relevant Information Element is specified in the IPFIX Information Model [I-D.ietf-ipfix-information-model-rfc5102bis], registered with IANA, or specified as an enterprise-specific Information Element. The specifications in the IPFIX protocol [I-D.ietf-ipfix-protocol-rfc5101bis] have not been defined in the context of an IPFIX Mediator receiving, aggregating, correlating, anonymizing, etc... Flow Records from the one or multiple Exporters. Indeed, the IPFIX protocol must be adapted for Intermediate Processes, as defined in the IPFIX Mediation Reference Model as specified in Figure A of [RFC6183], which is based on the IPFIX Mediation Problem Statement [RFC5982].

This document specifies the IP Flow Information Export (IPFIX) protocol in the context of the implementation and deployment of IPFIX Mediators. The use of the IPFIX protocol within a Mediator -- a device which contains both as a Collecting Process and an Exporting Process -- has an impact on the technical details of the usage of the protocol. An overview of the technical problem is covered in section 6 of [RFC5982]: loss of original exporter information, loss of base time information, transport sessions management, loss of Options Template Information, Template Id management, considerations for network considerations for aggregation.

The specifications in this document are based on the IPFIX protocol specifications [I-D.ietf-ipfix-protocol-rfc5101bis] but adapted according to the IPFIX Mediation Framework [RFC6183].

1.1. IPFIX Documents Overview

The IPFIX Protocol [I-D.ietf-ipfix-protocol-rfc5101bis] 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 the IPFIX Architecture [RFC5470], per the requirements defined in the IPFIX Requirement doc, [RFC3917].

The IPFIX Architecture [RFC5470] specifies how IPFIX Data Records and Templates are carried via a congestion-aware transport protocol from IPFIX Exporting Processes to IPFIX Collecting Processes.

IPFIX has a formal description of IPFIX Information Elements, their name, type and additional semantic information, as specified in the IPFIX Information Model [I-D.ietf-ipfix-information-model-rfc5102bis].

The IPFIX Applicability Statement [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.

"IPFIX Mediation: Problem Statement" [RFC5982], describing the IPFIX Mediation applicability examples, along with some problems that network administrators have been facing, is the basis for the "IPFIX Mediation: Framework" [RFC6183]. This framework details the IPFIX Mediation reference model and the components of an IPFIX Mediator.

1.2. IPFIX Mediator Documents Overview

The "IPFIX Mediation: Problem Statement" [RFC5982] provides an overview of the applicability of Mediators, and defines requirements for Mediators in general terms. This document is of use largely to define the problems to be solved through the deployment of IPFIX Mediators, and to provide scope to the role of Mediators within an IPFIX collection infrastructure.

The "IPFIX Mediation: Framework" [RFC6183] provides more architectural details of the arrangement of Intermediate Processes within a Mediator.

The details of specific Intermediate Processes, when these have additional export specifications (e.g., metadata about the intermediate processing conveyed through IPFIX Options Templates), are each treated in their own document (e.g., the "IP Flow Anonymization Support" [RFC6235]). Documents specifying the operations of specific Intermediate Processes cover the operation of these Processes within the Mediator framework, and comply with the specifications given in this document; they may additionally specify the operation of the process independently, outside the context of a Mediator, when this is appropriate. As of today, these documents are:

1. "IP Flow Anonymization Support", [RFC6235], which describes Anonymization techniques for IP flow data and the export of Anonymized data using the IPFIX protocol.

2. "Flow Selection Techniques" [I-D.ietf-ipfix-flow-selection-tech], which describes the process of selecting a subset of flows from all flows observed at an observation point, the flow selection motivations, and some specific flow selection techniques.
3. "Exporting Aggregated Flow Data using IP Flow Information Export" [I-D.ietf-ipfix-a9n] which describes Aggregated Flow export within the framework of IPFIX Mediators and defines an interoperable, implementation-independent method for Aggregated Flow export.

This document specifies the IP Flow Information Export (IPFIX) protocol specific to Mediation, i.e. the specifications that all Intermediate Processes type must comply to. Some extra specifications might be required per Intermediate Process type (In which case, the Intermediate Process specific document would cover those).

1.3. Relationship with IPFIX and PSAMP

The specification in this document applies to the IPFIX protocol specifications [I-D.ietf-ipfix-protocol-rfc5101bis]. All specifications from [I-D.ietf-ipfix-protocol-rfc5101bis] apply unless specified otherwise in this document.

As the Packet Sampling (PSAMP) protocol specifications [RFC5476] are based on the IPFIX protocol specifications, the specifications in this document are also valid for the PSAMP protocol. Therefore, the method specified by this document also applies to PSAMP.

2. Terminology

[EDITOR'S NOTE: change to only define terms in this section that are actually used in the document.]

[EDITOR'S NOTE: Definition change proposal for the Intermediate Process, Intermediate Conversion Process, Intermediate Selection Process, Intermediate Anonymization Process, and IPFIX Mediator. See <http://www.ietf.org/mail-archive/web/ipfix/current/msg05969.html>. However, the definitions are copied over verbatim from RFC6183. Also note that Intermediate Anonymization Process in this document is not in line with the RFC6235.]

IPFIX-specific terms, such as Observation Domain, Flow, Flow Key, Metering Process, Exporting Process, Exporter, IPFIX Device, Collecting Process, Collector, Template, IPFIX Message, Message Header, Template Record, Data Record, Options Template Record, Set,

Data Set, Information Element, and Transport Session, used in this document are defined in [I-D.ietf-ipfix-protocol-rfc5101bis]. The PSAMP-specific terms used in this document, such as Filtering and Sampling, are defined in [RFC5476].

IPFIX Mediation terms related to aggregation, such as the Interval, Aggregated Flow, and Aggregated Function are defined in [I-D.ietf-ipfix-a9n].

The IPFIX Mediation-specific terminology used in this document is defined in "IPFIX Mediation: Problem Statement" [RFC5982], and reused in "IPFIX Mediation: Framework" [RFC6183]. However, since both of those documents are informational RFCs, the definitions have been reproduced here along with additional definitions.

Similarly, since [RFC6235] is an experimental RFC, the Anonymization Record, Anonymized Data Record, and Intermediate Anonymization Process terms, specified in [RFC6235], are also reproduced here.

In this document, as in [I-D.ietf-ipfix-protocol-rfc5101bis], [RFC5476], [I-D.ietf-ipfix-a9n], and [RFC6235], the first letter of each IPFIX-specific and PSAMP-specific term is capitalized along with the IPFIX Mediation-specific term defined here. In this document, we call a stream of records carrying flow- or packet-based information a "record stream". The records may be encoded as IPFIX Data Records of any other format.

Transport Session Information: The Transport Session is specified in [I-D.ietf-ipfix-protocol-rfc5101bis]. In SCTP, the Transport Session Information is the SCTP association. In TCP and UDP, the Transport Session Information corresponds to a 5-tuple {Exporter IP address, Collector IP address, Exporter transport port, Collector transport port, transport protocol}.

Original Exporter: An Original Exporter is an IPFIX Device that hosts the Observation Points where the metered IP packets are observed.

Original Observation Point: An Observation Point of the Original Exporter. In the case of the Intermediate Aggregation Process on an IPFIX Mediator, the Original Observation Point can be composed of, but not limited to, a (set of) specific exporter(s), a (set of) specific interface(s) on an Exporter, a (set of) line card(s) on an Exporter, or any combinations of these.

IPFIX Mediation: IPFIX Mediation is the manipulation and conversion of a record stream for subsequent export using the IPFIX protocol.

Template Mapping: A mapping from Template Records and/or Options Template Records received by a Mediator to Template Records and/or Options Template Records sent by that IPFIX Mediator. Each entry in a Template Mapping is scoped by incoming or outgoing Transport Session and Observation Domain, as with Templates and Options Templates in the IPFIX Protocol.

Anonymization Record: A record that defines the properties of the anonymization applied to a single Information Element within a single Template or Options Template, as in [RFC6235].

Anonymized Data Record: A Data Record within a Data Set containing at least one Information Element with Anonymized values. The Information Element(s) within the Template or Options Template describing this Data Record SHOULD have a corresponding Anonymization Record, as in [RFC6235].

The following terms are used in this document to describe the architectural entities used by IPFIX Mediation.

Intermediate Process: An Intermediate Process takes a record stream as its input from Collecting Processes, Metering Processes, IPFIX File Readers, other Intermediate Processes, or other record sources; performs some transformations on this stream, based upon the content of each record, states maintained across multiple records, or other data sources; and passes the transformed record stream as its output to Exporting Processes, IPFIX File Writers, or other Intermediate Processes, in order to perform IPFIX Mediation. Typically, an Intermediate Process is hosted by an IPFIX Mediator. Alternatively, an Intermediate Process may be hosted by an Original Exporter.

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

Specific Intermediate Processes are described below. However, this is not an exhaustive list.

Intermediate Conversion Process: An Intermediate Conversion Process is an Intermediate Process that transforms non-IPFIX into IPFIX, or manages the relation among Templates and states of incoming/outgoing Transport Sessions (or equivalent for non IPFIX protocols) in the case of transport protocol conversion (e.g., from UDP to SCTP).

Intermediate Aggregation Process: An Intermediate Aggregation Process is an Intermediate Process that aggregates records based upon a set of Flow Keys or functions applied to fields from the record (e.g., binning and subnet aggregation).

Intermediate Correlation Process: An Intermediate Correlation Process is an Intermediate Process that adds information to records, noting correlations among them, or generates new records with correlated data from multiple records (e.g., the production of bidirectional flow records from unidirectional flow records).

Intermediate Selection Process: An Intermediate Selection Process is an Intermediate Process that selects records from a sequence based upon criteria-evaluated record values and passes only those records that match the criteria (e.g., Filtering only records from a given network to a given Collector).

Intermediate Anonymization Process: An Intermediate Anonymization Process is an Intermediate Process that transforms records in order to anonymize them, to protect the identity of the entities described by the records (e.g., by applying prefix-preserving pseudonymization of IP addresses).

3. Handling IPFIX Message Headers

The format of the IPFIX Message Header as exported by an IPFIX Mediator is shown in Figure 1. Note that the format is compatible with the IPFIX Message Header defined in [I-D.ietf-ipfix-protocol-rfc5101bis], with some field definitions (for the example, the Export Time) updated in the context of the IPFIX Mediator.

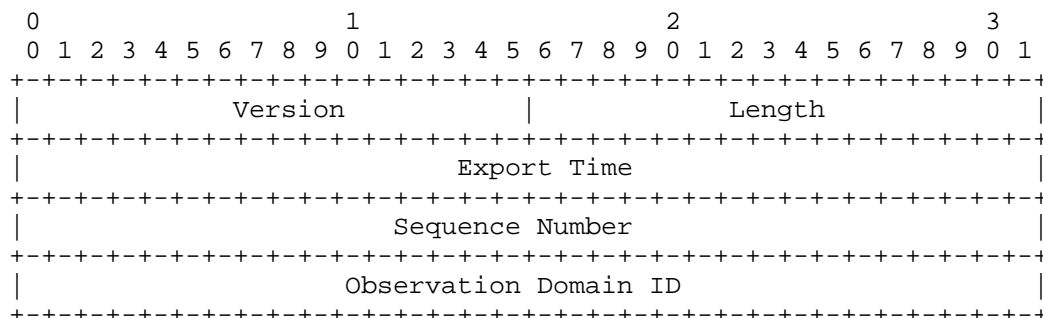


Figure 1: IP Message Header format

The header fields as exported by an IPFIX Mediator are describe below.

Version: Version of Flow Record format exported in this message. The value of this field is 0x000a for the current version, incrementing by one the version used in the NetFlow services export version 9 [RFC3954].

Length: Total length of the IPFIX Message, measured in octets, including Message Header and Set(s).

Export Time: Time at which the IPFIX Message leaves the Mediator, expressed in seconds since the UNIX epoch of 1 January 1970 at 00:00 UTC, as defined in [POSIX.1] encoded as an unsigned 32-bit integer. However, in the specific case of an IPFIX Mediator containing an Intermediate Conversion Process, the IPFIX Mediator MAY keep the export time received from the incoming Transport Session.

Sequence Number: Incremental sequence counter modulo 2^{32} of all IPFIX Data Records sent in a the current stream from the current Observation Domain by the Exporting Process. Each SCTP Stream counts sequence numbers separately, while all messages in a TCP connection or UDP transport session are considered to be part of the same stream. This value SHOULD be used by the Collecting Process to identify whether any IPFIX Data Records have been missed. Template and Options Template Records do not increase the Sequence Number.

Observation Domain ID: A 32-bit identifier of the Observation Domain that is locally unique to the Exporting Process. The Exporting Process uses the Observation Domain ID to uniquely identify to the Collecting Process the Observation Domain that metered the Flows. It is RECOMMENDED that this identifier should

be unique per IPFIX Device. Collecting Processes SHOULD use the Transport Session and the Observation Domain ID field to separate different export streams originating from the same Exporting Process. The Observation Domain ID SHOULD be 0 when no specific Observation Domain ID is relevant for the entire IPFIX Message. For example, when exporting the Exporting Process Statistics, or in case of hierarchy of Collector when aggregated Data Records are exported. See Section 4.1 for special considerations for Observation Domain management while passing unmodified templates through a Mediator, and Section 5 for guidelines for preservation of original Observation Domain information at a Mediator.

4. Template Management

How a Mediator handles the Templates it receives from the Original Exporter depends entirely on the nature of the Intermediate Process running on that Mediator. For Mediators which pass substantially the same Data Records from the Original Exporter downstream, (e.g., an Intermediate Selection Process), the templates can be passed unmodified as described in Section 4.1; this section describes a Template Mapping required to make this work in the general case. Mediators which export Data Records which are substantially changed from the Data Records received from the Original Exporter follow the guidelines in Section 4.1 instead.

Subsequent subsections deal with specific issues in Template management that may occur at Mediators.

4.1. Passing Unmodified Templates through a Mediator

[EDITOR'S NOTE: the definition of template mappings seems really implementation specific -- why not notionally just map IDs on each socket to a base template? on the other hand, if we're providing a real example, it should have concrete content in each field. reformatting is held off until this issue is resolved.]

The first case is a situation where the IPFIX Mediator doesn't modify the (Options) Template Record(s) content. A typical example is an Intermediate Selection Process acting as distributor, which collects Flow Records from one or more Exporters, and based on the Information Elements content, redirects the Flow Records to the appropriate Collector. This example is a typical case of a single network operation center managing multiple universities: an unique IPFIX Collector collects all Flow Records for the common infrastructure, but might be re-exporting specific university Flow Records to the responsible system administrator.

As specified in [I-D.ietf-ipfix-protocol-rfc5101bis], the Template IDs are unique per Exporter, per Transport Session, and per Observation Domain. As there is no guarantee that, for similar Template Records, the Template IDs received on the incoming Transport Session and exported to the outgoing Transport Session would be same, the IPFIX Mediator MUST maintain a Template Mapping composed of related received and exported (Options) Template Records:

- o for each received (Options) Template Record: Template Record Flow Keys and non Flow Keys, Template ID, Observation Domain Id, and Transport Session Information
- o for each exported (Options) Template Record: Template Record Flow Keys and non Flow Keys, Template ID, Collector, Observation Domain Id, and Transport Session Information

If an IPFIX Mediator receives an IPFIX Withdrawal Message for a (Options) Template Record that is not used anymore in any other Template Mappings, the IPFIX Mediator SHOULD export the appropriate IPFIX Withdrawal Message(s) on the outgoing Transport Session, and remove the corresponding entry in the Template Mapping.

If a (Options) Template Record is not used anymore in an outgoing Transport Session, it MUST be withdrawn with an IPFIX Template Withdrawal Message on that specific outgoing Transport Session, and its entry MUST be removed from the Template Mapping.

If an incoming or outgoing Transport Session is gracefully shutdown or reset, the (Options) Template Records corresponding to that Transport Session MUST be removed from the Template Mapping.

For example, Figure 2 displays an example of an Intermediate Selection Process, re-distributing Data Records to Collectors on the basis of customer networks, i.e. the Route Distinguisher (RD). In this example, the Template Record received from the Exporter #1 is reused towards Collector #1, Collector #2, and Collector #3.

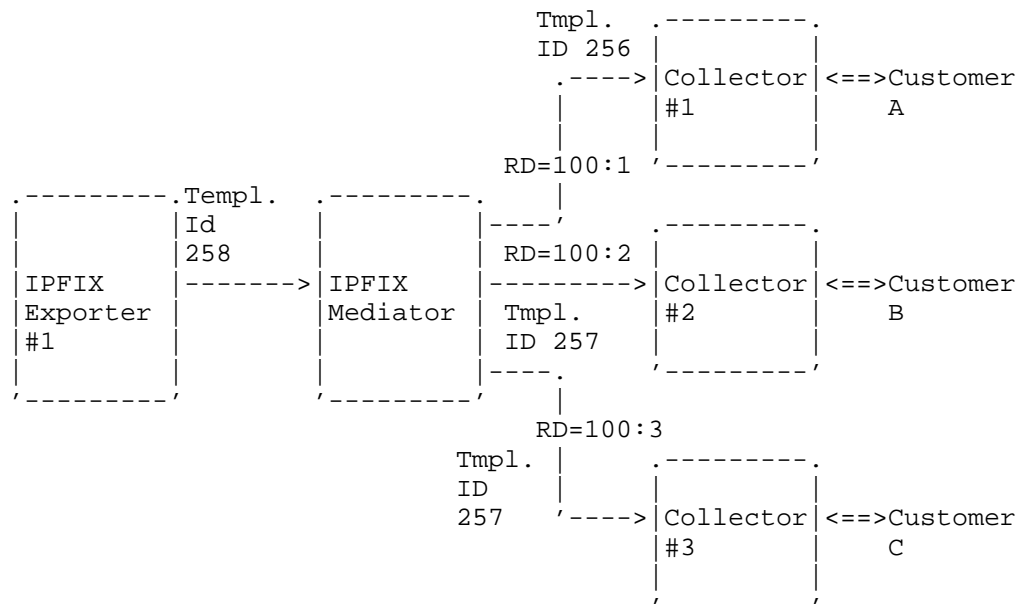


Figure 2: Intermediate Selection Process example

Figure 3 shows the Template Mapping for the system shown in Figure 2.

Template Entry A:

Incoming Transport Session Information (from Exporter#1):

Source IP: <Exporter#1 export IP address>

Destination IP: <IPFIX Mediator IP address>

Protocol: SCTP

Source Port: <source port>

Destination Port: 4739 (IPFIX)

Observation Domain Id: <Observation Domain ID>

Template Id: 258

Flow Keys: <series of Flow Keys>

Non Flow Keys: <series of non Flow Keys>

Template Entry B:

Outgoing Transport Session Information (to Collector#1):

Source IP: <IPFIX Mediator IP address>

Destination IP: <IPFIX Collector#1 IP address>

Protocol: SCTP

Source Port: <source port>

Destination Port: 4739 (IPFIX)

Observation Domain Id: <Observation Domain ID>

Template Id: 256

Flow Keys: <series of Flow Keys>

Non Flow Keys: <series of non Flow Keys>

Template Entry C:

Outgoing Transport Session Information (to Collector#2):

Source IP: <IPFIX Mediator IP address>

Destination IP: <IPFIX Collector#2 IP address>

Protocol: SCTP

Source Port: <source port>

Destination Port: 4739 (IPFIX)

Observation Domain Id: <Observation Domain ID>

Template Id: 257

Flow Keys: <series of Flow Keys>

Non Flow Keys: <series of non Flow Keys>

Template Entry D:

Outgoing Transport Session Information (to Collector#3):

Source IP: <IPFIX Mediator IP address>

Destination IP: <IPFIX Collector#3 IP address>

Protocol: SCTP

Source Port: <source port>

Destination Port: 4739 (IPFIX)

Observation Domain Id: <Observation Domain ID>

Template Id: 257

Flow Keys: <series of Flow Keys>

Non Flow Keys: <series of non Flow Keys>

Figure 3: Template Mapping example: templates

The Template Mapping corresponding to figure B can be displayed as:

```

Template Entry A  <----> Template Entry B
Template Entry A  <----> Template Entry C
Template Entry A  <----> Template Entry D

```

Template Mapping example: mappings

Alternatively, the Template Mapping may be optimized as:

```

          +--> Template Entry B
          |
Template Entry A  <--+--> Template Entry C
          |
          +--> Template Entry D

```

Template Mapping example: mappings

Note that all examples use Transport Sessions based on the SCTP protocol, as simplified use cases. However, the protocol would be important in situations such as an Intermediate Conversion Process doing transport protocol conversion.

4.2. Creating New Templates at a Mediator

The second case is a situation where the IPFIX Mediator generates new (Options) Template Records as a result of the Intermediate Process.

In this situation, the IPFIX Mediator doesn't need to maintain a Template Mapping, as it generates its own series of (Options) Template Records. However, the following special case might still require a Template Mapping, i.e. a situation where the IPFIX Mediator, typically containing an Intermediate Conversion Process, Intermediate Aggregation Process [I-D.ietf-ipfix-a9n], or Intermediate Anonymization Process in case of black-marker Anonymization [RFC6235], generates new (Options) Template Records based on what it receives from the Exporter(s), and based on the Intermediate Process function. In such a case, it's important to keep the correlation between the received (Options) Template Records and exported Derived (Options) Template Records in the Template Mapping. These template mappings would be kept as in Section 4.1, except that the export template would not be identical to the collection template.

4.3. Information Element Ordering within Templates

[EDITOR'S NOTE: address the following: What Paul Aikten would like to see in section 3.5 (See <http://www.ietf.org/mail-archive/web/ipfix/current/msg05969.html>): What about IE ordering? May an exporter re-order received fields? eg, two devices sending the same information, though with the fields in a different order. Or the mediator is extracting the same information from two sources. That seems to be a valid scenario. eg, this reduces the number of templates received at the collector.]

4.4. Handling Unknown Information Elements

[EDITOR'S NOTE: also from Paul Aitken: What should a mediator do with a field which it doesn't know/understand? Inevitably, exporters will be updated without mediators keeping in step. It's also very likely that mediators will see Enterprise-specific IEs. May a mediator re-export unknown IEs unchanged, or should it drop them? Presumably a mediator may report received Enterprise-specific IEs even from multiple different Enterprises. What if an unknown field depends on the field ordering? eg, it's a bitfield like flowKeyIndicator. Re-ordering, adding or removing fields breaks the meaning of this field, so it can't be passed on. It can only be used if the received fields are reported unchanged.]

5. Preserving Original Observation Point Information

[EDITOR'S NOTE: Decide whether we want to address export of observation point information without 6313. Review this section to make sure it adequately explains how original Observation Point information can get so complicated.]

Depending on the use case, the Collector in an Exporter - Mediator - Collector structure may need to receive information about the Original Observation Point(s), otherwise it may wrongly conclude that the IPFIX Device exporting the Flow Records, i.e. the IPFIX Mediator, directly observed the packets that generated the Flow Records. Two new Information Elements are introduced in the subsections below to address this use case: `originalExporterIPv4Address` and `originalExporterIPv6Address`. Practically, the Original Exporters will not export these Information Elements. Therefore, the Intermediate Process SHOULD report the Original Observation Point(s) to the best of its knowledge. Note that the Configuration Data Model for IPFIX and PSAMP [I-D.ietf-ipfix-configuration-model] may help.

In the IPFIX Mediator, the Observation Point(s) may be represented by:

- o A single Original Exporter (represented by the originalExporterIPv4Address or originalExporterIPv6Address Information Elements)
- o A list of Original Exporters (represented by the originalExporterIPv4Address or originalExporterIPv6Address Information Elements).
- o Any combination or list of Information Elements representing Observation Points. For example:
 - * A list of Original Exporter interface(s) (represented by the originalExporterIPv4Address or originalExporterIPv6Address, the ingressInterface and/or egressInterface Information Elements, respectively)
 - * A list of Original Exporter line card (represented by the originalExporterIPv4Address or originalExporterIPv6Address, the lineCardId Information Elements, respectively)

Some Information Elements characterizing the Observation Point may be added. For example, the flowDirection Information Element specifies the direction of the observation, and, as such, characterizes the Observation Point.

Any combination of the above representations is possible. For example, in case of an Intermediate Aggregation Process, an Original Observation Point could be composed of:

```
exporterIPv4Address 192.0.2.1
exporterIPv4Address 192.0.2.2,
  interface ethernet 0, direction ingress
  interface ethernet 1, direction ingress
  interface serial 1, direction egress
  interface serial 2, direction egress
exporterIPv4Address 192.0.2.3,
  lineCardId 1, direction ingress
```

Figure 4: Complex Observation Point Definition Example

If the Original Observation Point is composed of a list, then the IPFIX Structured Data [RFC6313] MUST be used to export it from the IPFIX Mediator.

The most generic way to export the Original Observation Point is to use a subTemplateMultiList, with the semantic "exactlyOneOf". Taking the previous example, the following encoding can be used:

Template Record 257: exporterIPv4Address
Template Record 258: exporterIPv4Address,
 basicList of ingressInterface, flowDirection
Template Record 259: exporterIPv4Address, lineCardId, flowDirection

Figure 5: Complex Observation Point Definition Example: Templates

The Original Observation Point is modeled with the Data Records corresponding to either Template Record 1, Template Record 2, or Template Record 3 but not more than one of these ("exactlyOneOf" semantic). This implies that the Flow was observed at exactly one of the Observation Points reported.

When an IPFIX Mediator receives Flow Records containing the Original Observation Point Information Element, i.e. originalExporterIPv6Address or originalExporterIPv4Address, the IPFIX Mediator SHOULD NOT modify its value(s) when composing new Flow Records in the general case. Known exceptions include anonymization per [RFC6235] section 7.2.4 and an Intermediate Correlation Process rewriting addresses across NAT. In other words, the Original Observation Point should not be replaced with the IPFIX Mediator Observation Point. The daisy chain of (Exporter, Observation Point) representing the path the Flow Records took from the Exporter to the top Collector in the Exporter - Mediator(s) - Collector structure model is out of the scope of this specification.

5.1. originalExporterIPv4Address Information Element

Description: The IPv4 address used by the Exporting Process on an Original Exporter, as seen by the Collecting Process on an IPFIX Mediator. Used to provide information about the Original Observation Points to a downstream Collector.

Data Type: ipv4Address

ElementId: TBD1

5.2. originalExporterIPv6Address Information Element

Description: The IPv6 address used by the Exporting Process on an Original Exporter, as seen by the Collecting Process on an IPFIX Mediator. Used to provide information about the Original Observation Points to a downstream Collector.

Data Type: ipv6Address

ElementId: TBD2

6. Managing Observation Domain IDs

In any case, the Observation Domain ID of any IPFIX Message containing Flow Records relevant to no particular Observation Domain, or to multiple Observation Domains, MUST have an Observation Domain ID of 0, as in Section 3 above, and section 3.1 of [I-D.ietf-ipfix-protocol-rfc5101bis].

IPFIX Mediators that do not change (Options) Template Records MUST maintain a Template Mapping, as detailed in Section 4.1, to ensure that the combination of Observation Domain IDs and Template IDs do not collide on export.

For IPFIX Mediators that export New (Options) Template Records, as in Section 4.2, there are two options for Observation Domain ID management. The first and simplest of these is to completely decouple exported Observation Domain IDs from received Observation Domain IDs; the IPFIX Mediator, in this case, comprises its own set of Observation Domain(s) independent of the Observation Domain(s) of the Original Exporters.

The second option is to provide or maintain a Template Mapping for received (Options) Template Records and exported inferred (Options) Template Records, along with the appropriate Observation Domain IDs per Transport Session, which ensures that the combination of Observation Domain IDs and Template IDs do not collide on export.

In some cases where the IPFIX Message Header can't contain a consistent Observation Domain for the entire IPFIX Message, but the Flow Records exported from the IPFIX Mediator should anyway contain the Observation Domain of the Original Exporter, the (Options) Template Record must contain the originalObservationDomainId Information Element. When an IPFIX Mediator receives Flow Records containing the originalObservationDomainId Information Element, the IPFIX Mediator MUST NOT modify its value(s) when composing new Flow Records with the originalObservationDomainId Information Element.

6.1. originalObservationDomainId Information Element

Description: The Observation Domain ID reported by the Exporting Process on an Original Exporter, as seen by the Collecting Process on an IPFIX Mediator. Used to provide information about the Original Observation Domain to a downstream Collector.

Data Type: unsigned32

Data Type Semantics: identifier

ElementId: TBD3

7. Timing Considerations

The IPFIX Message Header "Export Time" field is the time in seconds since 0000 UTC Jan 1, 1970, at which the IPFIX Message leaves the IPFIX Mediator. However, in the specific case of an IPFIX Mediator containing an Intermediate Conversion Process, the IPFIX Mediator MAY keep the export time received from the incoming Transport Session.

It is RECOMMENDED that Mediators handle time using absolute timestamps (e.g. flowStartSeconds, flowStartMilliseconds, flowStartNanoseconds), which are specified relative to the UNIX epoch (00:00 UTC 1 Jan 1970), where possible, rather than relative timestamps (e.g. flowStartSysUpTime, flowStartDeltaMicroseconds), which are specified relative to protocol structures such as system initialization or message export time.

The latter are difficult to manage for two reasons. First, they require constant translation, as the system initialization time of an intermediate system and the export time of an intermediate message will change across mediation operations. Further, relative timestamps introduce range problems. For example, when using the flowStartDeltaMicroseconds and flowEndDeltaMicroseconds Information Elements [iana-ipfix-assignments], the Data Record must be exported within a maximum of 71 minutes after its creation. Otherwise, the 32-bit counter would not be sufficient to contain the flow start time offset. Those time constraints might be incompatible with some of the application requirements of some Intermediate Processes.

Intermediate Processes MUST NOT assume that received records appear in flowStartTime, flowEndTime, or observationTime order. An Intermediate Process processing timing information (e.g., an Intermediate Aggregation Process) MAY ignore records that are significantly out of order, in order to meet application-specific state and latency requirements, but SHOULD report that records were dropped.

When an Intermediate Process aggregates information from different Flow Records, the timestamps on exported records SHOULD be the minimum of the start times and the maximum of the end times in the general case. However, if the Flow Records do not overlap, i.e. if there is a time gap between the times in the Flow Records, then the

report may be inaccurate. The IPFIX Mediator is only reporting what it knows, on the basis of the information made available to it - and there may not have been any data to observe during the gap. Then again, if there is an overlap in timestamps, there's the potential of double-accounting: different Observation Points may have observed the same traffic simultaneously. Therefore, as there is not a single rule that fits all different situations, a complete specification of the precise rules of applying Flow Record timestamps at IPFIX Mediators is out of the scope of this document.

Note that [I-D.ietf-ipfix-a9n] provides additional specifications for handling of timestamps at an Intermediate Aggregation Process.

8. Transport Considerations

SCTP [RFC4960] using the PR-SCTP extension specified in [RFC3758] MUST be implemented by all compliant IPFIX Mediator implementations. TCP [RFC0793] MAY also be implemented by IPFIX Mediator compliant implementations. UDP [RFC0768] MAY also be implemented by compliant IPFIX Mediator implementations. Transport-specific considerations for IPFIX Exporters as specified in sections 8.3, 8.4, 9.1, 9.2, and 10 of [I-D.ietf-ipfix-protocol-rfc5101bis] apply to IPFIX Mediators as well.

SCTP SHOULD be used in deployments where IPFIX Mediators and Collectors are communicating over links that are susceptible to congestion. SCTP is capable of providing any required degree of reliability. TCP MAY be used in deployments where IPFIX Mediators and Collectors communicate over links that are susceptible to congestion, but SCTP is preferred due to its ability to limit back pressure on Exporters and its message versus stream orientation. UDP MAY be used, although it is not a congestion-aware protocol. However, in this case, the IPFIX traffic between IPFIX Mediator and Collector MUST run in an environment where IPFIX traffic has been provisioned for, or is contained through some other means.

9. Collecting Process Considerations

Any Collecting Process compliant with [I-D.ietf-ipfix-protocol-rfc5101bis] can receive IPFIX Messages from an IPFIX Mediator. If the IPFIX Mediator uses IPFIX Structured Data [RFC6313] to export Original Exporter Information as in Section 5, the Collecting Process MUST support [RFC6313].

10. Specific Reporting Requirements

IPFIX provides Options Templates for the reporting on the reliability of processes within the IPFIX Architecture. As each Mediator includes at least one IPFIX Exporting Process, they SHOULD use the Exporting Process Reliability Statistics Options Template, as specified in [I-D.ietf-ipfix-protocol-rfc5101bis].

Analogous to the Metering Process Reliability Statistics Options Template, also specified in [I-D.ietf-ipfix-protocol-rfc5101bis], Mediators SHOULD implement the Intermediate Process Reliability Statistics Options Template, specified in the subsection below.

The Flow Keys Options Template, as specified in [I-D.ietf-ipfix-protocol-rfc5101bis], may require special handling at an IPFIX Mediator as described below.

In addition, each Intermediate Process may have its own specific reporting requirements (e.g. Anonymization Records as in [RFC6235], or the Aggregation Counter Distribution Options Template as in [I-D.ietf-ipfix-a9n]); these SHOULD be implemented as necessary as described in the specification for each Intermediate Process.

10.1. Intermediate Process Reliability Statistics Template

The Intermediate Process Statistics Options Template specifies the structure of a Data Record for reporting Intermediate Process statistics. It SHOULD contain the following Information Elements; the intermediateProcessId Information Element is defined in Section 10.3, and the ignoredRecordTotalCount Information Element is defined in Section 10.4:

IE	Description
observationDomainId [scope]	An identifier of the Observation Domain (of messages exported by this Mediator), locally unique to the Intermediate Process, to which this statistics record applies.
intermediateProcessId [scope]	An identifier for the Intermediate Process to which this statistics record applies.
ignoredRecordTotalCount	The total number of Data Records received but not processed by the Intermediate Process.

time first record ignored	The timestamp of the first record that was ignored by the Intermediate Process. For Data Records containing timestamp ranges, this SHOULD be taken from the start timestamp of the range; for data records containing no timing information, this SHOULD be taken from the Export Time in the message header of the containing IPFIX Message. For this timestamp, any of the following timestamp can be used: observationTimeSeconds, observationTimeMilliseconds, observationTimeMicroseconds, or observationTimeNanoseconds.
time last record ignored	The timestamp of the last record that was ignored by the Intermediate Process. For Data Records containing timestamp ranges, this SHOULD be taken from the end timestamp of the range; for data records containing no timing information, this SHOULD be taken from the Export Time in the message header of the containing IPFIX Message. For this timestamp, any of the following timestamp can be used: observationTimeSeconds, observationTimeMilliseconds, observationTimeMicroseconds, or observationTimeNanoseconds.

10.2. Flow Key Options Template

The Flow Keys Option Template specifies the structure of a Data Record for reporting the Flow Keys of reported Flows. A Flow Keys Data Record extends a particular Template Record that is referenced by its templateId identifier. The Template Record is extended by specifying which of the Information Elements contained in the corresponding Data Records describe Flow properties that serve as Flow Keys of the reported Flow. This Options Template is defined in section 4.4 of [I-D.ietf-ipfix-protocol-rfc5101bis], and SHOULD be used by Mediators for export as defined there.

When an Intermediate Process exports Data Records containing different Flow Keys from those received from the Original Exporter, and the Original Exporter sent a Flow Keys Options record to the Mediator, the Mediator MUST export a Flow Keys Options record

defining the the new set of Flow Keys.

10.3. intermediateProcessId Information Element

Description: An identifier of an Intermediate Process that is unique per IPFIX Device. Typically, this Information Element is used for limiting the scope of other Information Elements. Note that process identifiers may be assigned dynamically; ie., and Intermediate Process may be re-started with a different ID.

Data Type: unsigned32

Data Type Semantics: identifier

ElementId: TBD4

10.4. ignoredRecordTotalCount Information Element

Description: The total number of received Data Records that the Intermediate Process did not process since the (re-)initialization of the Intermediate Process; includes only Data Records not examined or otherwise handled by the Intermediate Process due to resource constraints, not Data Records which were examined or otherwise handled by the Intermediate Process but which merely do not contribute to any exported Data Record due to the operations performed by the Intermediate Process.

Data Type: unsigned64

Data Type Semantics: totalCounter

ElementId: TBD5

11. Configuration Management

In general, using Mediators to combine information from multiple Original Exporters requires a consistent configuration of the Metering Processes behind these Original Exporters. The details of this consistency are specific to each Intermediate Process. Consistency of configuration should be verified out of band, with the MIB modules ([I-D.ietf-ipfix-rfc5815bis] and [I-D.ietf-ipfix-psamp-mib]) or with the Configuration Data Model for IPFIX and PSAMP [I-D.ietf-ipfix-configuration-model]

12. Security Considerations

As they act as both IPFIX Collecting Processes and Exporting Processes, the Security Considerations for IPFIX Protocol [I-D.ietf-ipfix-protocol-rfc5101bis] also apply to Mediators. The Security Considerations for IPFIX Files [RFC5655] also apply to IPFIX Mediators that write IPFIX Files or use them for internal storage. However, there are a few specific considerations that IPFIX Mediator implementations must also take into account.

By design, IPFIX Mediators are "men-in-the-middle": they intercede in the communication between an Original Exporter (or another upstream Mediator) and a downstream Collecting Process. This has two important implications for the level of confidentiality provided across an IPFIX Mediator, and the ability to protect data integrity and Original Exporter authenticity across a Mediator. These are addressed in more detail in the Security Considerations for Mediators in [RFC6183].

Note that, while Mediators can use the exporterCertificate and collectorCertificate Information Elements defined in [RFC5655] as described in section 9.3 of [RFC6183] to export information about X.509 identities in upstream TLS-protected Transport Sessions, this mechanism cannot be used to provide true end-to-end assertions about a chain of IPFIX Mediators: any Mediator in the chain can simply falsify the information about upstream Transport Sessions. In situations where information about the chain of mediation is important, it must be determined out of band.

13. IANA Considerations

This document specifies n new IPFIX Information Elements, originalExporterIPv4Address in Section 5.1, originalExporterIPv6Address in Section 5.2, and originalObservationDomainId in Section 6.1, to be added to the IPFIX Information Element registry [iana-ipfix-assignments]. [IANA NOTE: please add the three Information Elements as specified in the references subsections, and change TBD1, TBD2, and TBD3 in this document to reflect the assigned identifiers.]

14. Acknowledgments

We would like to thank the IPFIX contributors, specifically Paul Aitken for his thorough review and Rahul Patel for his feedback and comments. This work is materially supported by the European Union Seventh Framework Programme under grant agreement 257315 (DEMONS).

15. References

15.1. Normative References

- [I-D.ietf-ipfix-protocol-rfc5101bis]
Claise, B. and B. Trammell, "Specification of the IP Flow Information eXport (IPFIX) Protocol for the Exchange of Flow Information", draft-ietf-ipfix-protocol-rfc5101bis-02 (work in progress), June 2012.
- [I-D.ietf-ipfix-information-model-rfc5102bis]
Claise, B. and B. Trammell, "Information Model for IP Flow Information eXport (IPFIX)", draft-ietf-ipfix-information-model-rfc5102bis-02 (work in progress), June 2012.
- [RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.
- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [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.
- [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.
- [RFC6313] Claise, B., Dhandapani, G., Aitken, P., and S. Yates, "Export of Structured Data in IP Flow Information Export (IPFIX)", RFC 6313, July 2011.
- [I-D.ietf-ipfix-flow-selection-tech]
D'Antonio, S., Zseby, T., Henke, C., and L. Peluso, "Flow Selection Techniques", draft-ietf-ipfix-flow-selection-tech-11 (work in progress), April 2012.
- [I-D.ietf-ipfix-a9n]
Trammell, B., Wagner, A., and B. Claise, "Flow Aggregation

for the IP Flow Information Export (IPFIX) Protocol",
draft-ietf-ipfix-a9n-05 (work in progress), July 2012.

[I-D.ietf-ipfix-psamp-mib]

Dietz, T., Claise, B., and J. Quittek, "Definitions of
Managed Objects for Packet Sampling",
draft-ietf-ipfix-psamp-mib-05 (work in progress),
July 2012.

[I-D.ietf-ipfix-configuration-model]

Muenz, G., Claise, B., and P. Aitken, "Configuration Data
Model for IPFIX and PSAMP",
draft-ietf-ipfix-configuration-model-11 (work in
progress), June 2012.

[I-D.ietf-ipfix-rfc5815bis]

Dietz, T., Kobayashi, A., Claise, B., and G. Muenz,
"Definitions of Managed Objects for IP Flow Information
Export", draft-ietf-ipfix-rfc5815bis-03 (work in
progress), March 2012.

15.2. Informative References

- [RFC3917] Quittek, J., Zseby, T., Claise, B., and S. Zander,
"Requirements for IP Flow Information Export (IPFIX)",
RFC 3917, October 2004.
- [RFC3954] Claise, B., "Cisco Systems NetFlow Services Export Version
9", RFC 3954, October 2004.
- [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.
- [RFC5476] Claise, B., Johnson, A., and J. Quittek, "Packet Sampling
(PSAMP) Protocol Specifications", RFC 5476, March 2009.
- [RFC5982] Kobayashi, A. and B. Claise, "IP Flow Information Export
(IPFIX) Mediation: Problem Statement", RFC 5982,
August 2010.
- [RFC6183] Kobayashi, A., Claise, B., Muenz, G., and K. Ishibashi,
"IP Flow Information Export (IPFIX) Mediation: Framework",
RFC 6183, April 2011.

[RFC6235] Boschi, E. and B. Trammell, "IP Flow Anonymization Support", RFC 6235, May 2011.

[iana-ipfix-assignments]
Internet Assigned Numbers Authority, "IP Flow Information
Export Information Elements
(<http://www.iana.org/assignments/ipfix/ipfix.xml>)".

[POSIX.1] IEEE, "IEEE 1003.1-2008 - IEEE Standard for Information
Technology - Portable Operating System Interface".

Authors' Addresses

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

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

Atsushi Kobayashi
NTT Information Sharing Platform Laboratories
3-9-11 Midori-cho
Musashino-shi, Tokyo 180-8585
Japan

Phone: +81 422 59 3978
Email: akoba@nttv6.net

Brian Trammell
Swiss Federal Institute of Technology Zurich
Gloriastrasse 35
8092 Zurich
Switzerland

Phone: +41 44 632 70 13
Email: trammell@tik.ee.ethz.ch

Network Working Group
Internet Draft
Obsoletes: 5101
Category: Standards Track
Expires: December 30, 2012

B. Claise, Ed.
Cisco Systems, Inc.
B. Trammell, Ed.
ETH Zurich
June 28, 2012

Specification of the IP Flow Information eXport (IPFIX) Protocol
for the Exchange of Flow Information
draft-ietf-ipfix-protocol-rfc5101bis-02

Abstract

This document specifies the IP Flow Information Export (IPFIX) protocol that serves for transmitting Traffic Flow information over the network. In order to transmit Traffic Flow information from an Exporting Process to an information Collecting Process, a common representation of flow data and a standard means of communicating them is required. This document describes how the IPFIX Data and Template Records are carried over a number of transport protocols from an IPFIX Exporting Process to an IPFIX Collecting Process. This document obsoletes RFC 5101.

Status of This Memo

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

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

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

This Internet-Draft will expire on March 23, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	5
1.1. Changes since RFC 5101	5
1.2. IPFIX Documents Overview	6
2. Terminology	6
2.1. Terminology Summary Table	11
3. IPFIX Message Format	12
3.1. Message Header Format	14
3.2. Field Specifier Format	15
3.3. Set and Set Header Format	16
3.3.1. Set Format	16
3.3.2. Set Header Format	17
3.4. Record Format	18
3.4.1. Template Record Format	18
3.4.2. Options Template Record Format	20
3.4.2.1. Scope	21
3.4.2.2. Options Template Record Format	22
3.4.3. Data Record Format	24
4. Specific Reporting Requirements	25
4.1. The Metering Process Statistics Options Template	25
4.2. The Metering Process Reliability Statistics Options Template	26
4.3. The Exporting Process Reliability Statistics Options Template	28
4.4. The Flow Keys Options Template	29
5. IPFIX Message Header Export Time and Flow Record Time	30
6. Linkage with the Information Model	30
6.1. Encoding of IPFIX Data Types	31
6.1.1. Integral Data Types	31
6.1.2. Address Types	31
6.1.3. float32	31
6.1.4. float64	31
6.1.5. boolean	31
6.1.6. string and octetArray	31
6.1.7. dateTimeSeconds	31
6.1.8. dateTimeMilliseconds	32

6.1.9	dateTimeMicroseconds	32
6.1.10	dateTimeNanoseconds	32
6.2.	Reduced Size Encoding	33
7.	Variable-Length Information Element	34
8.	Template Management	36
8.1.	Template Withdrawal and Redefinition	37
8.2	Sequencing Template Management Actions	39
8.3.	Additional considerations for Template Management over Sctp	40
8.4.	Additional considerations for Template Management over UDP	40
9.	The Collecting Process's Side	41
9.1.	Additional considerations for Sctp Collecting Processes	42
9.2.	Additional considerations for UDP Collecting Processes	42
10.	Transport Protocol	42
10.1.	Transport Compliance and Transport Usage	44
10.2.	Sctp	44
10.2.1.	Congestion Avoidance	44
10.2.2.	Reliability	44
10.2.3.	MTU	45
10.2.4.	Association Establishment and Shutdown	45
10.2.5.	Failover	46
10.2.6.	Streams	46
10.3.	UDP	46
10.3.1.	Congestion Avoidance	46
10.3.2.	Reliability	46
10.3.3.	MTU	47
10.3.4.	Session Establishment and Shutdown	47
10.3.5.	Failover and Session Duplication	47
10.4.	Tcp	48
10.4.1.	Congestion Avoidance	48
10.4.2.	Reliability	49
10.4.3.	MTU	49
10.4.4.	Connection Establishment, Shutdown, and Restart	49
10.4.5.	Failover	50
11.	Security Considerations	50
11.1.	Applicability of TLS and DTLS	51
11.2.	Usage	52
11.3.	Authentication	52
11.4.	Protection against DoS Attacks	53
11.5.	When DTLS or TLS Is Not an Option	54
11.6.	Logging an IPFIX Attack	55
11.7.	Securing the Collector	55
12.	IANA Considerations	55
Appendix A.	IPFIX Encoding Examples	56
A.1.	Message Header Example	56
A.2.	Template Set Examples	57
A.2.1.	Template Set Using IETF-Specified Information	

Elements	57
A.2.2. Template Set Using Enterprise-Specific Information	
Elements	57
A.3. Data Set Example	59
A.4. Options Template Set Examples	60
A.4.1. Options Template Set Using IETF-Specified	
Information Elements	60
A.4.2. Options Template Set Using Enterprise-Specific	
Information	60
A.4.3. Options Template Set Using an Enterprise-Specific	
Scope	61
A.4.4. Data Set Using an Enterprise-Specific Scope	62
A.5. Variable-Length Information Element Examples	63
A.5.1. Example of Variable-Length Information Element with	
Length	63
A.5.2. Example of Variable-Length Information Element with	
3 Octet Length Encoding	63
References	63
Normative References	63
Informative References	64
Acknowledgments	66
Authors' Addresses	67

OPEN ISSUES:

RFC2026 section 4.1.2: "The requirement for at least two independent and interoperable implementations applies to all of the options and features of the specification. In cases in which one or more options or features have not been demonstrated in at least two interoperable implementations, the specification may advance to the Draft Standard level only if those options or features are removed."

The interop report from Prague is at
<http://www.ietf.org/proceedings/80/slides/ipfix-4.pdf>

The following features have not yet been successfully interop'd; the document may have to be held pending successful interoperability testing

1. DTLS over SCTP (section 11.1)
2. DTLS over UDP (section 11.1)
3. multiple-stream export in SCTP (section 10.2.6)
4. Template withdrawal (section 8.1)
5. Template ID reuse (section 8.1)
6. Template stream separation (section 8.3)
7. Template expiration in UDP (section 8.4)

1. Introduction

Traffic on a data network can be seen as consisting of flows passing through network elements. It is often interesting, useful, or even necessary to have access to information about these flows that pass through the network elements for administrative or other purposes. A collecting process should be able to receive the flow information passing through multiple network elements within the data network. This requires uniformity in the method of representing the flow information and the means of communicating the flows from the network elements to the collection point. This document specifies a protocol to achieve these aforementioned requirements. This document specifies in detail the representation of different flows, the additional data required for flow interpretation, packet format, transport mechanisms used, security concerns, etc.

1.1. Changes since RFC 5101

This document obsoletes the Proposed Standard revision of the IPFIX Protocol Specification [RFC5101]. The protocol specified by this document is interoperable with the protocol as specified in [RFC5101]. The following changes have been made to this document with respect to the previous document:

- EDITOR'S NOTE: not sure if we need to this information

- Errata ID: 1655 (technical)
- Errata ID: 2791 (technical)
- Errata ID: 2814 (editorial)
- Errata ID: 1818 (editorial)
- Errata ID: 2792 (editorial)
- Errata ID: 2888 (editorial)
- Errata ID: 2889 (editorial)
- Errata ID: 2890 (editorial)
- Errata ID: 2891 (editorial)
- Errata ID: 2892 (editorial)
- Errata ID: 2903 (editorial)
- Errata ID: 2761 (editorial)
- Errata ID: 2762 (editorial)
- Errata ID: 2763 (editorial)
- Errata ID: 2764 (editorial)
- Errata ID: 2852 (editorial)
- Errata ID: 2857 (editorial)

- The encoding of the `dateTimeSeconds`, `dateTimeMilliseconds`, `dateTimeMicroseconds`, and `dateTimeNanoseconds` data types, and the related encoding of the IPFIX Message Header Export Time field, have been clarified, especially with respect to the epoch upon which the timestamp data types are based.

- Template management in section 8 has been simplified, and made as independent of transport protocol as is practically possible, by relaxing restrictions on template management actions.
- Editorial changes, including structural changes to sections 8, 9, and 10 to improve readability.

1.2. IPFIX Documents Overview

The IPFIX protocol 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]. This document specifies how IPFIX data records and templates are carried via a number of transport protocols from IPFIX Exporting Processes to IPFIX Collecting Processes.

Four IPFIX optimizations/extensions are currently specified: a bandwidth saving method for the IPFIX protocol in [RFC5473], an efficient method for exporting bidirectional flow in [RFC5103], a method for the definition and export of complex data structures in [RFC6313], and the specification of the Protocol for IPFIX Mediations [IPFIX-MED-PROTO] based on the IPIFX Mediation Framework [RFC6183].

IPFIX has a formal description of IPFIX Information Elements, their name, type and additional semantic information, as specified in [RFC5102bis], with the export of the Information Element types specified in [RFC5610].

[IPFIX-CONF] specifies a data model for configuring and monitoring IPFIX and PSAMP compliant devices using the NETCONF protocol, while the [RFC5815bis] specifies a MIB module for monitoring.

In terms of development, [RFC5153] provides guidelines for the implementation and use of the IPFIX protocol, while [RFC5471] provides guidelines for testing.

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.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

The definitions of the basic terms like Traffic Flow, Exporting Process, Collecting Process, Observation Points, etc. are semantically identical to those found in the IPFIX requirements document [RFC3917]. Some of the terms have been expanded for more clarity when defining the protocol. Additional terms required for the protocol have also been defined. Definitions in this document and in [RFC5470] are equivalent, except that definitions that are only relevant to the IPFIX protocol only appear here.

The terminology summary table in Section 2.1 gives a quick overview of the relationships between some of the different terms defined.

Observation Point

An Observation Point is a location in the network where packets can be observed. Examples include: a line to which a probe is attached, a shared medium, such as an Ethernet-based LAN, a single port of a router, or a set of interfaces (physical or logical) of a router.

Note that every Observation Point is associated with an Observation Domain (defined below), and that one Observation Point may be a superset of several other Observation Points. For example, one Observation Point can be an entire line card. That would be the superset of the individual Observation Points at the line card's interfaces.

Observation Domain

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

Traffic Flow or Flow

There are several definitions of the term 'flow' being used by the Internet community. Within the context of IPFIX we use the following definition:

A Flow is defined as a set of packets passing an Observation Point

in the network during a certain time interval. All packets belonging to a particular Flow have a set of common properties. Each property is defined as the result of applying a function to the values of:

1. one or more packet header fields (e.g., destination IP address), transport header fields (e.g., destination port number), or application header fields (e.g., RTP header fields [RFC3550]).
2. one or more characteristics of the packet itself (e.g., number of MPLS labels, etc...).
3. one or more of fields derived from packet treatment (e.g., next hop IP address, the output interface, etc...).

A packet is defined as belonging to a Flow if it completely satisfies all the defined properties of the Flow.

Note that the set of packets represented by a Flow may be empty; that is, a Flow may represent zero or more packets. Note also that as sampling is a packet treatment, this definition includes packets selected by a sampling mechanism.

Flow Key

Each of the fields that:

1. belong to the packet header (e.g., destination IP address),
2. are a property of the packet itself (e.g., packet length),
3. are derived from packet treatment (e.g., Autonomous System (AS) number),

and that are used to define a Flow are termed Flow Keys.

Flow Record

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

Metering Process

The Metering Process generates Flow Records. Inputs to the

process are packet headers and characteristics observed at an Observation Point, and packet treatment at the Observation Point (for example, the selected output interface).

The Metering Process consists of a set of functions that includes packet header capturing, timestamping, sampling, classifying, and maintaining Flow Records.

The maintenance of Flow Records may include 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.

Exporting Process

The Exporting Process sends Flow Records to one or more Collecting Processes. The Flow Records are generated by one or more Metering Processes.

Exporter

A device that hosts one or more Exporting Processes is termed an Exporter.

IPFIX Device

An IPFIX Device hosts at least one Exporting Process. It may host further Exporting Processes and arbitrary numbers of Observation Points and Metering Processes.

Collecting Process

A Collecting Process receives Flow Records from one or more Exporting Processes. The Collecting Process might process or store received Flow Records, but such actions are out of scope for this document.

Collector

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

Template

A Template is an ordered sequence of <type, length> pairs used to completely specify the structure and semantics of a particular set of information that needs to be communicated from an IPFIX Device to a Collector. Each Template is uniquely identifiable by means

of a Template ID.

IPFIX Message

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

Message Header

The Message Header is the first part of an IPFIX Message, which provides basic information about the message, such as the IPFIX version, length of the message, message sequence number, etc.

Template Record

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

Data Record

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

Options Template Record

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

Set

Set is a generic term for a collection of records that have a similar structure. In an IPFIX Message, one or more Sets follow the Message Header.

There are three different types of Sets: Template Set, Options Template Set, and Data Set.

Template Set

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

Options Template Set

An Options Template Set is a collection of one or more Options

Template Records that have been grouped together in an IPFIX Message.

Data Set

A Data Set is one or more Data Records, of the same type, that are grouped together in an IPFIX Message. Each Data Record is previously defined by a Template Record or an Options Template Record.

Information Element

An Information Element is a protocol and encoding-independent description of an attribute that may appear in an IPFIX Record. The IPFIX information model [RFC5102bis] defines the base set of Information Elements for IPFIX. The type associated with an Information Element indicates constraints on what it may contain and also determines the valid encoding mechanisms for use in IPFIX.

Transport Session

In Stream Control Transmission Protocol (SCTP), the transport session is known as the SCTP association, which is uniquely identified by the SCTP endpoints [RFC4960]; in TCP, the transport session is known as the TCP connection, which is uniquely identified by the combination of IP addresses and TCP ports used. In UDP, the transport session is known as the UDP session, which is uniquely identified by the combination of IP addresses and UDP ports used.

2.1. Terminology Summary Table

Set	contents	
	Template	record
Data Set	/	Data Record(s)
Template Set	Template Record(s)	/
Options Template Set	Options Template Record(s)	/

Figure A: Terminology Summary Table

A Data Set is composed of Data Record(s). No Template Record is included. A Template Record or an Options Template Record defines the Data Record.

A Template Set contains only Template Record(s).

An Options Template Set contains only Options Template Record(s).

3. IPFIX Message Format

An IPFIX Message consists of a Message Header, followed by one or more Sets. The Sets can be any of the possible three types: Data Set, Template Set, or Options Template Set.

The format of the IPFIX Message is shown in Figure B.



Figure B: IPFIX Message Format

The Exporter MUST code all binary integers of the Message Header and the different Sets in network-byte order (also known as the big-endian byte ordering).

Following are some examples of IPFIX Messages:

1. An IPFIX Message consisting of interleaved Template, Data, and Options Template Sets -- A newly created Template is exported as soon as possible. So, if there is already an IPFIX Message with a Data Set that is being prepared for export, the Template and Options Template Sets are interleaved with this information, subject to availability of space.

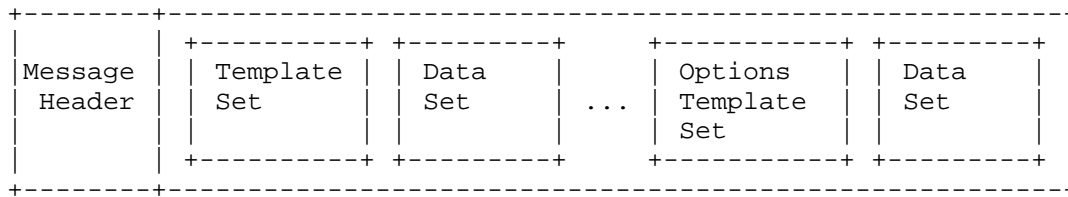


Figure C: IPFIX Message, Example 1

2. An IPFIX Message consisting entirely of Data Sets -- After the appropriate Template Records have been defined and transmitted to the Collecting Process, the majority of IPFIX Messages consist solely of Data Sets.

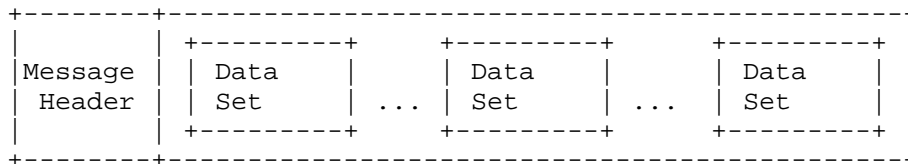


Figure D: IPFIX Message, Example 2

3. An IPFIX Message consisting entirely of Template and Options Template Sets.

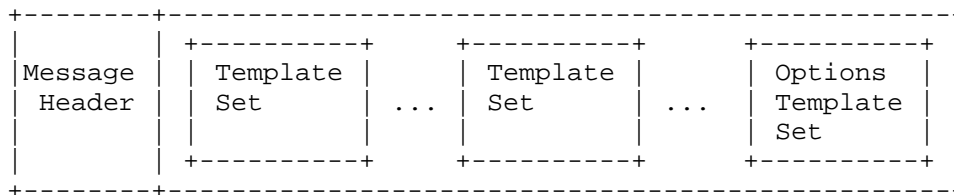


Figure E: IPFIX Message, Example 3

3.1. Message Header Format

The format of the IPFIX Message Header is shown in Figure F.

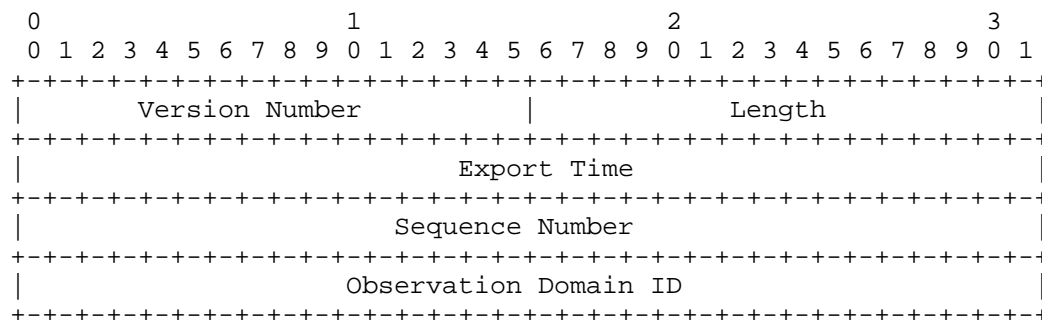


Figure F: IPFIX Message Header Format

Message Header Field Descriptions:

Version

Version of Flow Record format exported in this message. The value of this field is 0x000a for the current version, incrementing by one the version used in the NetFlow services export version 9 [RFC3954].

Length

Total length of the IPFIX Message, measured in octets, including Message Header and Set(s).

Export Time

Time at which the IPFIX Message Header leaves the Exporter, expressed in seconds since the UNIX epoch of 1 January 1970 at 00:00 UTC, encoded as an unsigned 32-bit integer.

Sequence Number

Incremental sequence counter modulo 2^{32} of all IPFIX Data Records sent on this SCTP stream from the current Observation Domain by the Exporting Process. Check the specific meaning of this field in the subsections of Section 10 when UDP or TCP is selected as the transport protocol. This value SHOULD be used by the Collecting Process to identify whether any IPFIX Data Records have been missed. Template and Options Template Records do not increase the Sequence Number.

Observation Domain ID

A 32-bit identifier of the Observation Domain that is locally unique to the Exporting Process. The Exporting Process uses the Observation Domain ID to uniquely identify to the Collecting Process the Observation Domain that metered the Flows. It is RECOMMENDED that this identifier also be unique per IPFIX Device. Collecting Processes SHOULD use the Transport Session and the Observation Domain ID field to separate different export streams originating from the same Exporter. The Observation Domain ID SHOULD be 0 when no specific Observation Domain ID is relevant for the entire IPFIX Message, for example, when exporting the Exporting Process Statistics, or in case of a hierarchy of Collectors when aggregated Data Records are exported.

3.2. Field Specifier Format

Vendors need the ability to define proprietary Information Elements, because, for example, they are delivering a pre-standards product, or the Information Element is, in some way, commercially sensitive. This section describes the Field Specifier format for both IETF-specified Information Elements [RFC5102bis] and enterprise-specific Information Elements.

The Information Elements are identified by the Information Element identifier. When the Enterprise bit is set to 0, the corresponding Information Element identifier will report an IETF-specified Information Element, and the Enterprise Number MUST NOT be present. When the Enterprise bit is set to 1, the corresponding Information Element identifier will report an enterprise-specific Information Element; the Enterprise Number MUST be present. An example of this is shown in Section A.4.2.

The Field Specifier format is shown in Figure G.

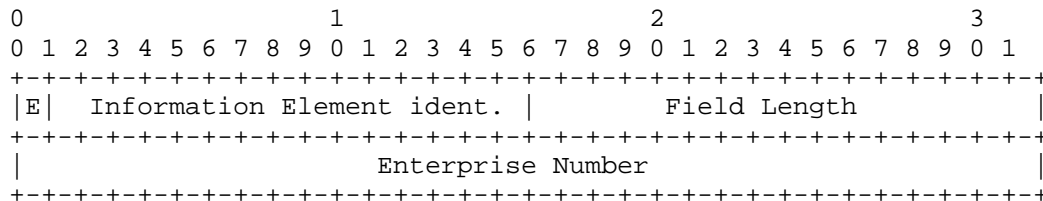


Figure G: Field Specifier Format

Where:

E

Enterprise bit. This is the first bit of the Field Specifier. If this bit is zero, the Information Element Identifier identifies an IETF-specified Information Element, and the four-octet Enterprise Number field MUST NOT be present. If this bit is one, the Information Element identifier identifies an enterprise-specific Information Element, and the Enterprise Number field MUST be present.

Information Element identifier

A numeric value that represents the type of Information Element. Refer to [RFC5102bis].

Field Length

The length of the corresponding encoded Information Element, in octets. Refer to [RFC5102bis]. The field length may be smaller than the definition in [RFC5102bis] if the reduced size encoding is used (see Section 6.2). The value 65535 is reserved for variable-length Information Elements (see Section 7).

Enterprise Number

IANA enterprise number [PEN] of the authority defining the Information Element identifier in this Template Record.

3.3. Set and Set Header Format

A Set is a generic term for a collection of records that have a similar structure. There are three different types of Sets: Template Sets, Options Template Sets, and Data Sets. Each of these Sets consists of a Set Header and one or more records. The Set Format and the Set Header Format are defined in the following sections.

3.3.1. Set Format

A Set has the format shown in Figure H. The record types can be either Template Records, Options Template Records, or Data Records. The record types MUST NOT be mixed within a Set.



Figure H: Set Format

The Set Field Definitions are as follows:

Set Header

The Set Header Format is defined in Section 3.3.2.

Record

One of the record Formats: Template Record, Options Template Record, or Data Record Format.

Padding

The Exporting Process MAY insert some padding octets, so that the subsequent Set starts at an aligned boundary. For security reasons, the padding octet(s) MUST be composed of zero (0) valued octets. The padding length MUST be shorter than any allowable record in this Set. If padding of the IPFIX Message is desired in combination with very short records, then the padding Information Element 'paddingOctets' [RFC5102bis] can be used for padding records such that their length is increased to a multiple of 4 or 8 octets. Because Template Sets are always 4-octet aligned by definition, padding is only needed in case of other alignments e.g., on 8-octet boundaries.

3.3.2. Set Header Format

Every Set contains a common header. This header is defined in Figure I.

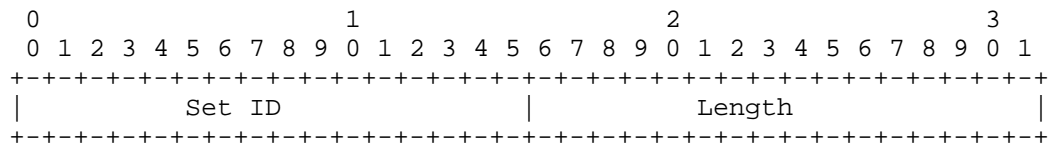


Figure I: Set Header Format

The Set Header Field Definitions are as follows:

Set ID

Set ID value identifies the Set. A value of 2 is reserved for the Template Set. A value of 3 is reserved for the Options Template Set. All other values from 4 to 255 are reserved for future use. Values above 255 are used for Data Sets. The Set ID values of 0 and 1 are not used for historical reasons [RFC3954].

Length

Total length of the Set, in octets, including the Set Header, all records, and the optional padding. Because an individual Set MAY contain multiple records, the Length value MUST be used to determine the position of the next Set.

3.4. Record Format

IPFIX defines three record formats, defined in the next sections: the Template Record Format, the Options Template Record Format, and the Data Record Format.

3.4.1. Template Record Format

One of the essential elements in the IPFIX record format is the Template Record. Templates greatly enhance the flexibility of the record format because they allow the Collecting Process to process IPFIX Messages without necessarily knowing the interpretation of all Data Records. A Template Record contains any combination of IANA-assigned and/or enterprise-specific Information Elements identifiers.

The format of the Template Record is shown in Figure J. It consists of a Template Record Header and one or more Field Specifiers. The definition of the Field Specifiers is given in Figure G above.

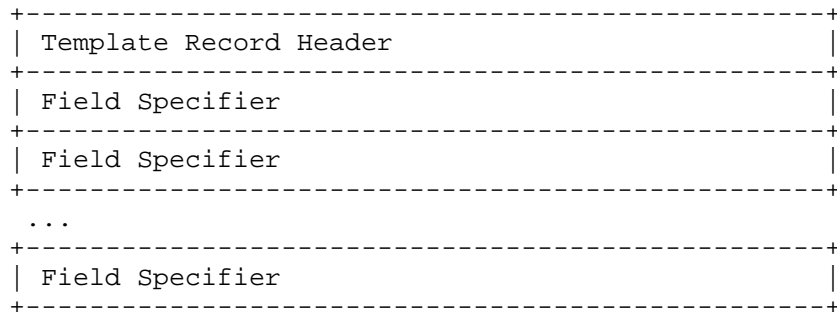


Figure J: Template Record Format

The format of the Template Record Header is shown in Figure K.

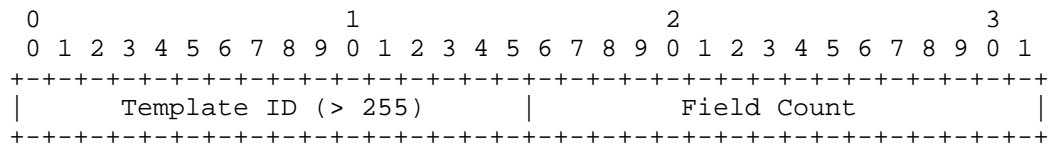


Figure K: Template Record Header Format

The Template Record Header Field Definitions are as follows:

Template ID

Each of the newly generated Template Records is given a unique Template ID. This uniqueness is local to the Transport Session and Observation Domain that generated the Template ID. Template IDs 0-255 are reserved for Template Sets, Options Template Sets, and other reserved Sets yet to be created. Template IDs of Data Sets are numbered from 256 to 65535. There are no constraints regarding the order of the Template ID allocation.

Field Count

Number of fields in this Template Record.

The example in Figure L shows a Template Set with mixed standard and enterprise-specific Information Elements. It consists of a Set Header, a Template Header, and several Field Specifiers.

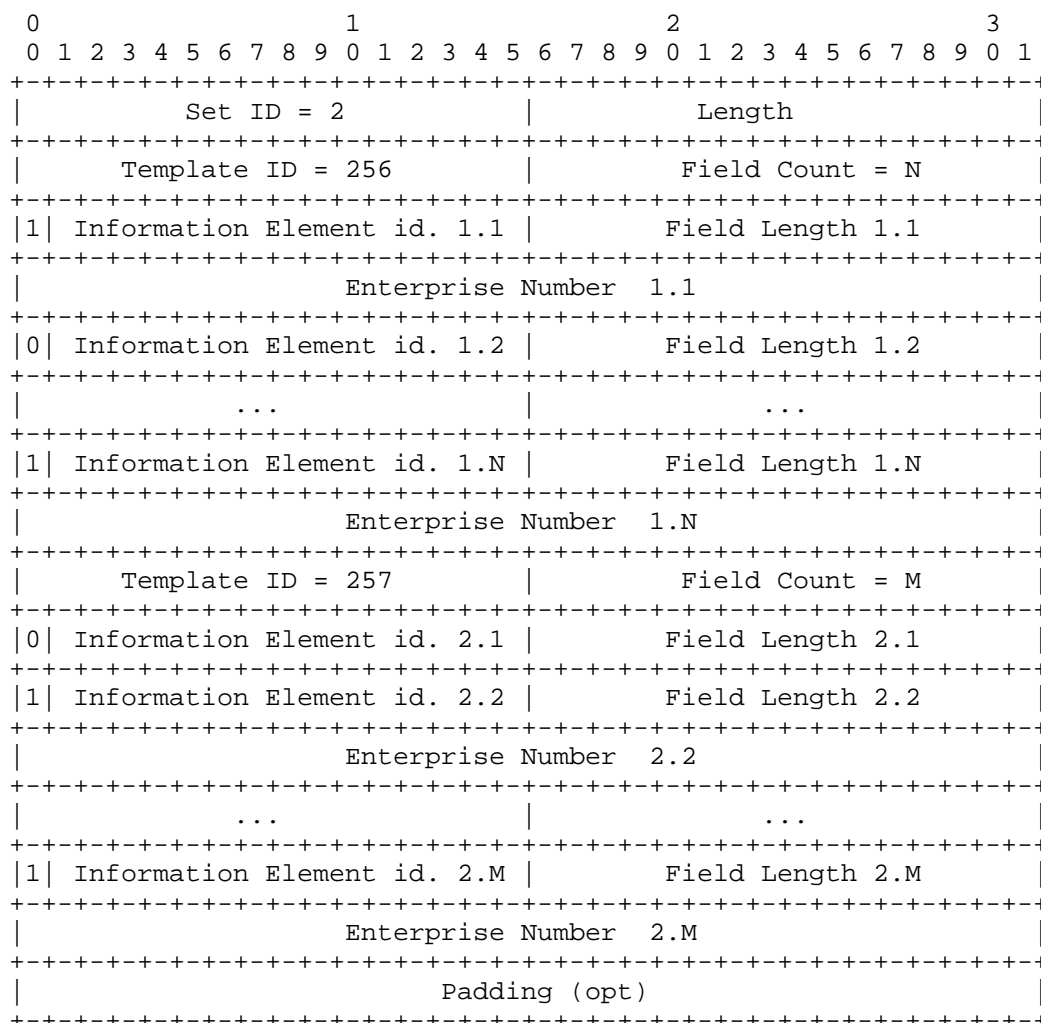


Figure L: Template Set Example

Information Element Identifiers 1.2 and 2.1 are defined by the IETF (Enterprise bit = 0) and, therefore, do not need an Enterprise Number to identify them.

3.4.2. Options Template Record Format

Thanks to the notion of scope, The Options Template Record gives the Exporter the ability to provide additional information to the Collector that would not be possible with Flow Records alone.

One Options Template Record example is the "Flow Keys", which reports the Flow Keys for a Template, which is defined as the scope. Another example is the "Template configuration", which reports the configuration sampling parameter(s) for the Template, which is defined as the scope.

3.4.2.1. Scope

The scope, which is only available in the Options Template Set, gives the context of the reported Information Elements in the Data Records.

Note that the IPFIX Message Header already contains the Observation Domain ID (the identifier of the Observation Domain). If not zero, this Observation Domain ID can be considered as an implicit scope for the Data Records in the IPFIX Message. The Observation Domain ID MUST be zero when the IPFIX Message contains Data Records with different Observation Domain ID values defined as scopes.

Multiple Scope Fields MAY be present in the Options Template Record, in which case, the composite scope is the combination of the scopes. For example, if the two scopes are defined as "metering process" and "template", the combined scope is this Template for this Metering Process. The order of the Scope Fields, as defined in the Options Template Record, is irrelevant in this case. However, if the order of the Scope Fields in the Options Template Record is relevant, the order of the Scope Fields MUST be used. For example, if the first scope defines the filtering function, while the second scope defines the sampling function, the order of the scope is important. Applying the sampling function first, followed by the filtering function, would lead to potentially different Data Records than applying the filtering function first, followed by the sampling function. In this case, the Collector deduces the function order by looking at the order of the scope in the Options Template Record.

The scope is an Information Element specified in the IPFIX Information Model [RFC5102bis]. An IPFIX-compliant implementation of the Collecting Process SHOULD support this minimum set of Information Elements as scope: LineCardId, TemplateId, exporterIPv4Address, exporterIPv6Address, and ingressInterface. Note that other Information Elements, such as meteringProcessId, exportingProcessId, observationDomainId, etc. are also valid scopes. The IPFIX protocol doesn't prevent the use of any Information Elements for scope. However, some Information Element types don't make sense if specified as scope; for example, the counter Information Elements.

Finally, note that the Scope Field Count MUST NOT be zero.

3.4.2.2. Options Template Record Format

An Options Template Record contains any combination of IANA-assigned and/or enterprise-specific Information Elements identifiers.

The format of the Options Template Record is shown in Figure M. It consists of an Options Template Record Header and one or more Field Specifiers. The definition of the Field Specifiers is given in Figure G above.

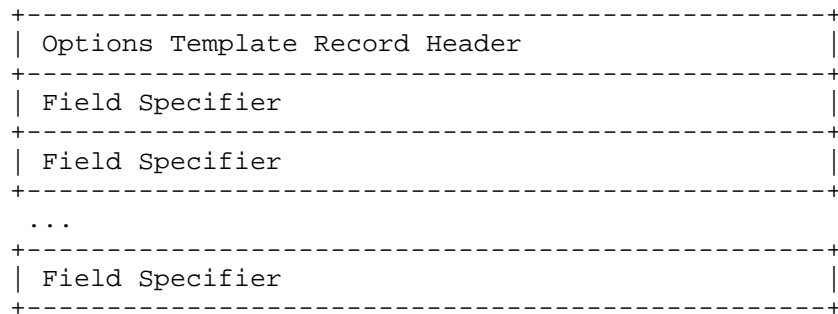


Figure M: Options Template Record Format

The format of the Options Template Record Header is shown in Figure N.

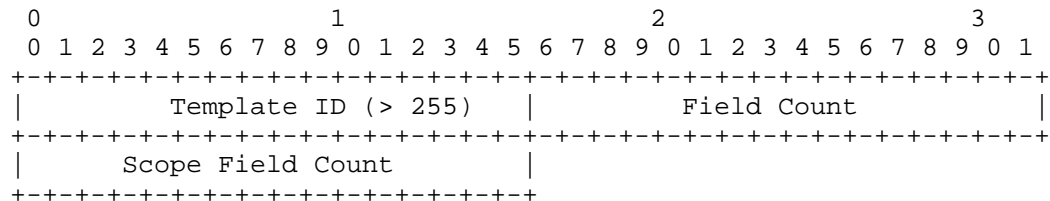


Figure N: Options Template Record Header Format

The Options Template Record Header Field Definitions are as follows:

Template ID

Template ID of this Options Template Record. This value is greater than 255.

Field Count

Number of all fields in this Options Template Record, including the Scope Fields.

Scope Field Count

Number of scope fields in this Options Template Record. The Scope Fields are normal Fields except that they are interpreted as scope at the Collector. The Scope Field Count MUST NOT be zero.

The example in Figure 0 shows an Options Template Set with mixed IETF and enterprise-specific Information Elements. It consists of a Set Header, an Options Template Header, and several Field Specifiers.

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Set ID = 3           |           Length           |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Template ID = 258    |           Field Count = N + M   |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Scope Field Count = N   | 0 | Scope 1 Infor. Element Id. |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Scope 1 Field Length    | 0 | Scope 2 Infor. Element Id. |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Scope 2 Field Length    |           ...           |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           ...                     | 1 | Scope N Infor. Element Id. |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Scope N Field Length    |           Scope N Enterprise Number ...
+-----+-----+-----+-----+-----+-----+-----+-----+
... Scope N Enterprise Number      | 1 | Option 1 Infor. Element Id. |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Option 1 Field Length    |           Option 1 Enterprise Number ...
+-----+-----+-----+-----+-----+-----+-----+-----+
... Option 1 Enterprise Number      |           ...           |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           ...                     | 0 | Option M Infor. Element Id. |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Option M Field Length    |           Padding (optional)   |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Figure 0: Options Template Set Example

3.4.3. Data Record Format

The Data Records are sent in Data Sets. The format of the Data Record is shown in Figure P. It consists only of one or more Field Values. The Template ID to which the Field Values belong is encoded in the Set Header field "Set ID", i.e., "Set ID" = "Template ID".

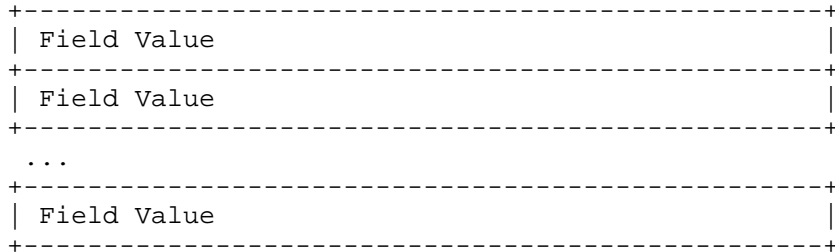


Figure P: Data Record Format

Note that Field Values do not necessarily have a length of 16 bits. Field Values are encoded according to their data type specified in [RFC5102bis].

Interpretation of the Data Record format can be done only if the Template Record corresponding to the Template ID is available at the Collecting Process.

The example in Figure Q shows a Data Set. It consists of a Set Header and several Field Values.

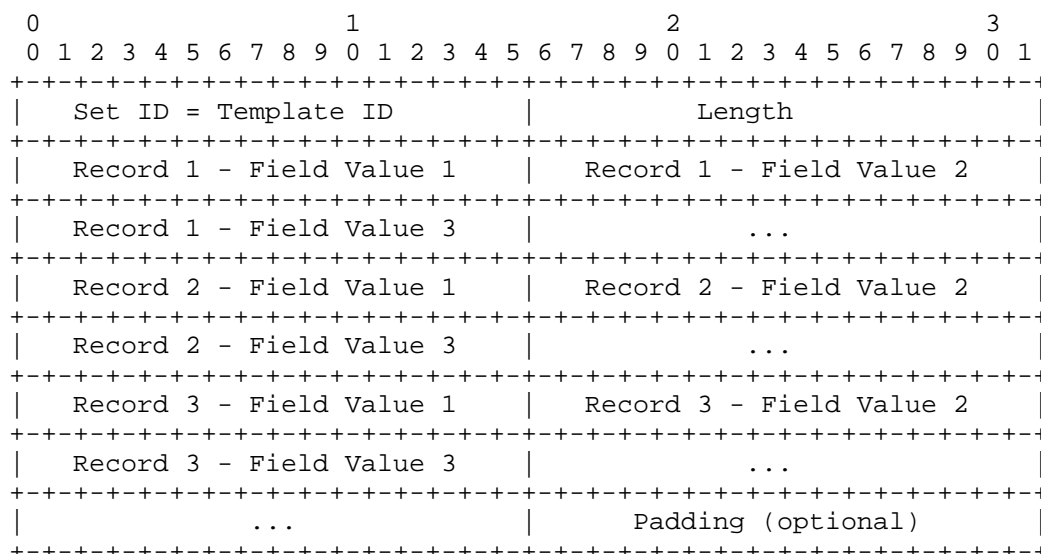


Figure Q: Data Set, Containing Data Records

4. Specific Reporting Requirements

Some specific Options Templates and Options Template Records are necessary to provide extra information about the Flow Records and about the Metering Process.

The Options Template and Options Template Records defined in these subsections, which impose some constraints on the Metering Process and Exporting Process implementations, MAY be implemented. If implemented, the specific Options Templates SHOULD be implemented as specified in these subsections.

The minimum set of Information Elements is always specified in these Specific IPFIX Options Templates. Nevertheless, extra Information Elements may be used in these specific Options Templates.

The Collecting Process MUST check the possible combinations of Information Elements within the Options Template Records to correctly interpret the following Options Templates.

4.1. The Metering Process Statistics Options Template

The Metering Process Statistics Options Template specifies the structure of a Data Record for reporting Metering Process statistics.

It SHOULD contain the following Information Elements that are defined in [RFC5102bis]:

(scope) observationDomainId

An identifier of an Observation Domain that is locally unique to the Exporting Process. This Information Element MUST be defined as a Scope Field.

(scope) meteringProcessId

An identifier of the Metering Process for which statistics are reported. This Information Element MUST be defined as a Scope Field.

exportedMessageTotalCount

The total number of IPFIX Messages that the Exporting Process successfully sent to the Collecting Process since the Exporting Process re-initialization.

exportedFlowRecordTotalCount

The total number of Flow Records that the Exporting Process successfully sent to the Collecting Process since the Exporting Process re-initialization.

exportedOctetTotalCount

The total number of octets that the Exporting Process successfully sent to the Collecting Process since the Exporting Process re-initialization.

The Exporting Process SHOULD export the Data Record specified by the Metering Process Statistics Options Template on a regular basis or based on some export policy. This periodicity or export policy SHOULD be configurable.

Note that if several Metering Processes are available on the Exporter Observation Domain, the Information Element meteringProcessId MUST be specified as an additional Scope Field.

4.2. The Metering Process Reliability Statistics Options Template

The Metering Process Reliability Options Template specifies the structure of a Data Record for reporting lack of reliability in the Metering Process. It SHOULD contain the following Information Elements that are defined in [RFC5102bis]:

(scope) observationDomainId

An identifier of an Observation Domain that is locally unique to the Exporting Process. This Information Element MUST be defined as a Scope Field.

(scope) meteringProcessId

The identifier of the Metering Process for which lack of reliability is reported. This Information Element MUST be defined as a Scope Field.

ignoredPacketTotalCount

The total number of IP packets that the Metering Process did not process.

ignoredOctetTotalCount

The total number of octets in observed packets that the Metering Process did not process.

time first packet ignored

The timestamp of the first packet that was ignored by the Metering Process. For this timestamp, any of the following timestamp can be used: observationTimeSeconds, observationTimeMilliseconds, observationTimeMicroseconds, or observationTimeNanoseconds.

time last packet ignored

The timestamp of the last packet that was ignored by the Metering Process. For this timestamp, any of the following timestamp can be used: observationTimeSeconds, observationTimeMilliseconds, observationTimeMicroseconds, or observationTimeNanoseconds.

The Exporting Process SHOULD export the Data Record specified by the Metering Process Reliability Statistics Options Template on a regular basis or based on some export policy. This periodicity or export policy SHOULD be configurable.

Note that if several Metering Processes are available on the Exporter Observation Domain, the Information Element meteringProcessId MUST be specified as an additional Scope Field.

Since the Metering Process Reliability Option Template will logically contain two identical timestamp Information Elements, and since the order of the Information Elements in the Template Records is not guaranteed, the Collecting Process MUST determine which is the oldest and the most recent timestamp in order to determine the right semantic behind the time first packet ignored and time last packet ignored Information Elements. Note that the counters wrap-around for the timestamps SHOULD also be taken into account.

4.3. The Exporting Process Reliability Statistics Options Template

The Exporting Process Reliability Options Template specifies the structure of a Data Record for reporting lack of reliability in the Exporting process. It SHOULD contain the following Information Elements that are defined in [RFC5102bis]:

(scope) Exporting Process ID

The identifier of the Exporting Process for which lack of reliability is reported. There are three Information Elements specified in [RFC5102bis] that can be used for this purpose: exporterIPv4Address, exporterIPv6Address, or exportingProcessId. This Information Element MUST be defined as a Scope Field.

notSentFlowTotalCount

The total number of Flows that were generated by the Metering Process and dropped by the Metering Process or by the Exporting Process instead of being sent to the Collecting Process.

notSentPacketTotalCount

The total number of packets in Flow Records that were generated by the Metering Process and dropped by the Metering Process or by the Exporting Process instead of being sent to the Collecting Process.

notSentOctetTotalCount

The total number of octets in packets in Flow Records that were generated by the Metering Process and dropped by the Metering Process or by the Exporting Process instead of being sent to the Collecting Process.

time first flow dropped

The time at which the first Flow Record was dropped by the Exporting Process. For this timestamp, any of the following timestamp can be used: observationTimeSeconds, observationTimeMilliseconds, observationTimeMicroseconds, or observationTimeNanoseconds.

time last flow dropped

The time at which the last Flow Record was dropped by the Exporting Process. For this timestamp, any of the following timestamp can be used: observationTimeSeconds, observationTimeMilliseconds, observationTimeMicroseconds, or observationTimeNanoseconds.

The Exporting Process SHOULD export the Data Record specified by the Exporting Process Reliability Statistics Options Template on a regular basis or based on some export policy. This periodicity or export policy SHOULD be configurable.

Since the Exporting Process Reliability Option Template will logically contain two identical timestamp Information Elements, and since the order of the Information Elements in the Template Records is not guaranteed, the Collecting Process MUST determine which is the oldest and the most recent timestamp in order to determine the right semantic behind the time first packet ignored and time last packet ignored Information Elements. Note that the counters wrap-around for the timestamps SHOULD also be taken into account.

4.4. The Flow Keys Options Template

The Flow Keys Options Template specifies the structure of a Data Record for reporting the Flow Keys of reported Flows. A Flow Keys Data Record extends a particular Template Record that is referenced by its templateId identifier. The Template Record is extended by specifying which of the Information Elements contained in the corresponding Data Records describe Flow properties that serve as Flow Keys of the reported Flow.

The Flow Keys Options Template SHOULD contain the following Information Elements that are defined in [RFC5102bis]:

(scope) templateId	An identifier of a Template. This Information Element MUST be defined as a Scope Field.
flowKeyIndicator	Bitmap with the positions of the Flow Keys in the Data Records.

5. IPFIX Message Header Export Time and Flow Record Time

The IPFIX Message Header Export Time field is the time at which the IPFIX Message Header leaves the Exporter, expressed in seconds since the UNIX epoch, 1 January 1970 at 00:00 UTC, encoded in an unsigned 32-bit integer.

Certain time-related Information Elements may be expressed as an offset from this Export Time. For example, Data Records requiring a microsecond precision can export the flow start and end times with the flowStartMicroseconds and flowEndMicroseconds Information Elements [RFC5102bis], which encode the absolute time in microseconds in terms of the NTP epoch, 1 January 1900 at 00:00 UTC, in a 64-bit field. An alternate solution is to export the flowStartDeltaMicroseconds and flowEndDeltaMicroseconds Information Elements [RFC5102bis] in the Data Record, which respectively report the flow start and end time as negative offsets from the Export Time, as an unsigned 32-bit integer. This latter solution lowers the export bandwidth requirement, saving two bytes per timestamp, while increasing the load on the Exporter, as the Exporting Process must calculate the flowStartDeltaMicroseconds and flowEndDeltaMicroseconds of every single Data Record before exporting the IPFIX Message.

It must be noted that timestamps based on the Export Time impose some time constraints on the Data Records contained within the IPFIX Message. In the example of flowStartDeltaMicroseconds and flowEndDeltaMicroseconds Information Elements [RFC5102bis], the Data Record can only contain records with timestamps within 71 minutes of the Export Time. Otherwise, the 32-bit counter would not be sufficient to contain the flow start time offset.

6. Linkage with the Information Model

The Information Elements [RFC5102bis] MUST be sent in canonical format in network-byte order (also known as the big-endian byte ordering).

6.1. Encoding of IPFIX Data Types

The following sections will define the encoding of the data types specified in [RFC5102bis].

6.1.1. Integral Data Types

Integral data types -- `octet`, `signed8`, `unsigned16`, `signed16`, `unsigned32`, `signed32`, `signed64`, and `unsigned64` -- MUST be encoded using the default canonical format in network-byte order. Signed Integral data types are represented in two's complement notation.

6.1.2. Address Types

Address types -- `macAddress`, `ipv4Address`, and `ipv6Address` -- MUST be encoded the same way as the integral data types. The `macAddress` is treated as a 6-octet integer, the `ipv4Address` as a 4-octet integer, and the `ipv6Address` as a 16-octet integer.

6.1.3. float32

The `float32` data type MUST be encoded as an IEEE single-precision 32-bit floating point-type, as specified in [IEEE.754.1985].

6.1.4. float64

The `float64` data type MUST be encoded as an IEEE double-precision 64-bit floating point-type, as specified in [IEEE.754.1985].

6.1.5. boolean

The `boolean` data type is specified according to the `TruthValue` in [RFC2579]: it is an integer with the value 1 for true and a value 2 for false. Every other value is undefined. The `boolean` data type MUST be encoded in a single octet.

6.1.6. string and octetArray

The data type `string` represents a finite length string of valid characters of the Unicode character encoding set. The `string` data type MUST be encoded in UTF-8 format. The string is sent as an array of octets using an Information Element of fixed or variable length. The length of the Information Element specifies the length of the `octetArray`.

6.1.7. dateTimeSeconds

The data type `dateTimeSeconds` is an unsigned 32 bit integer

containing the number of seconds since the UNIX epoch, 1 January 1970 at 00:00 UTC, as defined in [POSIX.1]. `dateTimeSeconds` is encoded identically to the IPFIX Message Header Export Time field. It can represent dates between 1 January 1970 and 8 February 2106.

6.1.8. `dateTimeMilliseconds`

The data type `dateTimeMilliseconds` is an unsigned 64-bit integer containing the number of milliseconds since the UNIX epoch, 1 January 1970 at 00:00 UTC, as defined in [POSIX.1]. It can represent dates beginning on 1 January 1970 for approximately the next 500 billion years.

6.1.9 `dateTimeMicroseconds`

The data type `dateTimeMicroseconds` is a 64-bit field encoded according to the NTP Timestamp format as defined in section 6 of [RFC5905]. This field is made up of two unsigned 32-bit integers, Seconds and Fraction. The Seconds field is the number of seconds since the NTP epoch, 1 January 1900 at 00:00 UTC. The Fraction field is the fractional number of seconds in units of $1/(2^{32})$ seconds (approximately 233 picoseconds). It can represent dates beginning between 1 January 1900 and 8 February 2036.

Note that `dateTimeMicroseconds` and `dateTimeNanoseconds` share an identical encoding. The `dateTimeMicroseconds` data type is intended only to represent timestamps of microsecond precision. Therefore, the bottom 11 bits of the fraction field MAY contain any value and MUST be ignored for all Information Elements of this data type (as $2^{11} \times 233$ picoseconds = .477 microseconds).

6.1.10 `dateTimeNanoseconds`

The data type `dateTimeNanoseconds` is a 64-bit field encoded according to the NTP Timestamp format as defined in section 6 of [RFC5905]. This field is made up of two unsigned 32-bit integers, Seconds and Fraction. The Seconds field is the number of seconds since the NTP epoch, 1 January 1900 at 00:00 UTC. The Fraction field is the fractional number of seconds in units of $1/(2^{32})$ seconds (approximately 233 picoseconds). It can represent dates beginning between 1 January 1900 and 8 February 2036.

Note that `dateTimeMicroseconds` and `dateTimeNanoseconds` share an identical encoding. There is no restriction on the interpretation of the Fraction field for the `dateTimeNanoseconds` data type.

6.2. Reduced Size Encoding

Information Elements encoded as signed, unsigned, or float data types MAY be encoded using fewer octets than those implied by their type in the information model definition [RFC5102bis], based on the assumption that the smaller size is sufficient to carry any value the Exporter may need to deliver. This reduces the network bandwidth requirement between the Exporter and the Collector. Note that the Information Element definitions [RFC5102bis] will always define the maximum encoding size.

For instance, the information model [RFC5102bis] defines `octetDeltaCount` as an `unsigned64` type, which would require 64 bits. However, if the Exporter will never locally encounter the need to send a value larger than 4294967295, it may chose to send the value instead as an `unsigned32`. For example, a core router would require an `unsigned64` `byteCount`, while an `unsigned32` might be sufficient for an access router.

This behavior is indicated by the Exporter by specifying a size in the Template with a smaller length than that associated with the assigned type of the Information Element. In the example above, the Exporter would place a length of 4 versus 8 in the Template.

If reduced size encoding MAY be be applied to the following integer types: `unsigned64`, `signed64`, `unsigned32`, `signed32`, `unsigned16`, and `signed16`. The signed versus unsigned property of the reported value MUST be preserved. The reduction in size can be to any number of octets smaller than the original type if the data value still fits, i.e., so that only leading zeroes are dropped. For example, an `unsigned64` can be reduced in size to 7, 6, 5, 4, 3, 2, or 1 octet(s).

Reduced size encoding MAY be used to reduce `float64` to `float32`. The `float32` not only has a reduced number range, but due to the smaller mantissa, is also less precise. In this case, the `float64` would be reduced in size to 4 octets.

Reduced size encoding MUST NOT be applied to any other data type defined in [RFC5102bis] that implies a fixed length, as these types either have internal structure (such as `ipv4Address` or `dateTimeMicroseconds`) or restricted ranges that are not suitable for reduced length encoding (such as `dateTimeMilliseconds`).

Information Elements of type `octetArray` and `string` may be exported using any length, subject to restrictions on length specific to each Information Element, as noted in that Information Element's description.

7. Variable-Length Information Element

The IPFIX Template mechanism is optimized for fixed-length Information Elements [RFC5102bis]. Where an Information Element has a variable length, the following mechanism **MUST** be used to carry the length information for both the IETF and proprietary Information Elements.

In the Template Set, the Information Element Field Length is recorded as 65535. This reserved length value notifies the Collecting Process that length of the Information Element will be carried in the Information Element content itself.

In most cases, the length of the Information Element will be less than 255 octets. The following length-encoding mechanism optimizes the overhead of carrying the Information Element length in this majority case. The length is carried in the octet before the Information Element, as shown in Figure R.

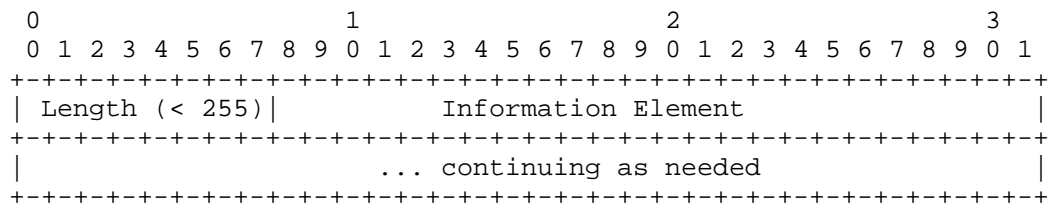


Figure R: Variable-Length Information Element (length < 255 octets)

The length may also be encoded into 3 octets before the Information element allowing the length of the Information Element to be greater than or equal to 255 octets. In this case, first octet of the Length field **MUST** be 255, and the length is carried in the second and third octets, as shown in Figure S.

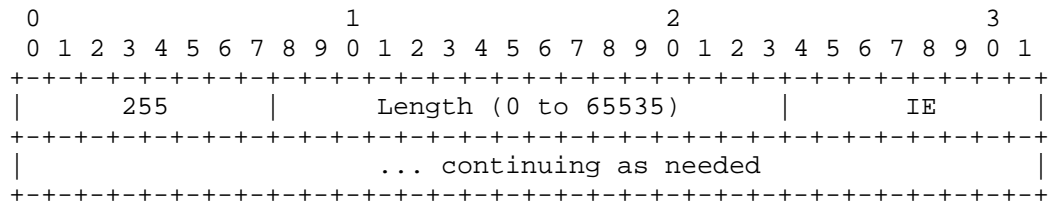


Figure S: Variable-Length Information Element (length 0 to 65535 octets)

The octets carrying the length (either the first or the first three octets) **MUST NOT** be included in the length of the Information

Element.

8. Template Management

This section describes the management of Templates and Options Templates at the Exporting and Collecting Processes. The goal of Template management is to ensure, to the extent possible, that the Exporting Process and Collecting Process have a consistent view of the Templates and Options Templates used to encode and decode the Records sent from the Exporting Process to the Collecting Process. Achieving this goal is complicated somewhat by two factors: 1. the need to support the reuse of Template IDs within a Transport Session and 2. the need to support unreliable transmission for templates when UDP is used as the transport protocol for IPFIX Messages.

The Template Management mechanisms defined in this section apply to IPFIX Message export on any supported Transport Protocol. Additional considerations specific to SCTP and UDP transport are given in sections 8.3 and 8.4, respectively.

The Exporting Process assigns and maintains the Template IDs per Transport Session for the Exporter's Observation Domains. A newly created Template Record is assigned an unused Template ID by the Exporting Process. The Collecting Process MUST store all received Template Record information for the duration of each Transport Session until reuse or withdrawal as in section 8.1, except as noted in section 8.4, so that it can interpret the corresponding Data Records that are received in subsequent Data Sets. The Collecting Process MUST NOT assume that the Template IDs from a given Exporting Process refer to the same Templates as they did in previous Transport Sessions from the same Exporting Process. When a Transport Session is closed, the Collecting Process MUST discard all Templates received over that association and stop decoding IPFIX Messages that use those Templates.

If a specific Information Element is required by a Template, but is not present in observed packets, the Exporting Process MAY choose to export Flow Records without this Information Element in a Data Record defined by a new Template.

If an Information Element is required more than once in a Template, the different occurrences of this Information Element SHOULD follow the logical order of their treatments by the Metering Process. For example, if a selected packet goes through two hash functions, and if the two hash values are sent within a single Template, the first occurrence of the hash value should belong to the first hash function in the Metering Process. For example, when exporting the two source IP addresses of an IPv4 in IPv4 packets, the first sourceIPv4Address Information Element occurrence should be the IPv4 address of the outer header, while the second occurrence should be the address of

the inner header. Collecting processes MUST properly handle Templates with multiple identical Information Elements.

The Exporting Process SHOULD transmit the Template Set and Options Template Set in advance of any Data Sets that use that (Options) Template ID, to help ensure that the Collector has the Template Record before receiving the first Data Record. Data Records that correspond to a Template Record MAY appear in the same and/or subsequent IPFIX Message(s).

This ensures that the Collecting Process normally receives Template Records from the Exporting Process before receiving Data Records. However, if the Template Records have not been received at the time Data Records are received, the Collecting Process MAY store the Data Records for a short period of time and decode them after the Template Records are received. In any case, a Collecting Process MUST NOT assume that the Data Set and the associated Template Set (or Options Template Set) are exported in the same IPFIX Message.

Different Observation Domains from the same Transport Session MAY use the same Template ID value to refer to different Templates; Collecting Processes MUST properly handle this case.

Options Templates and Templates which are related or interdependent (e.g. by sharing common properties as in [RFC5473]) SHOULD be sent together in the same IPFIX Message.

8.1. Template Withdrawal and Redefinition

Since a Template may have a lifetime at the Exporting Process independent of the Transport Session, IPFIX provides a mechanism for the withdrawal of templates and for the reuse of template IDs. This mechanism does not apply when UDP is used to transport IPFIX messages; for this case, see Section 8.4.

Templates that will not be used further by an Exporting Process MUST be withdrawn by sending a Template Withdrawal Message. After receiving a Template Withdrawal, a Collecting Process MUST discard the Template and stop using it to interpret Data Sets.

A Template Withdrawal consists of a Template Record for the Template ID to be withdrawn with a Field Count of 0. The format of a Template Withdrawal is shown in Figure T.

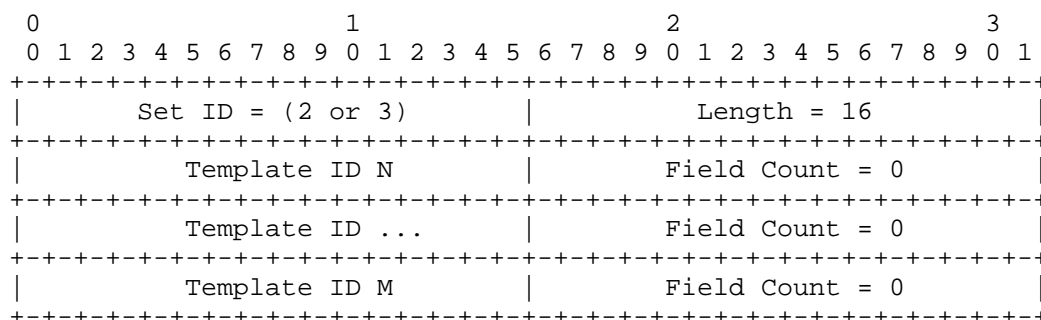


Figure T: Template Withdrawal Set Format

The Set ID field MUST contain the value 2 for Template Set Withdrawal and the value 3 for Options Template Set Withdrawal. Multiple Template IDs MAY be withdrawn with a single Template Withdrawal, in that case, padding MAY be used.

A Template Withdrawal Message is an IPFIX Message containing Template Withdrawals. It withdraws Template IDs for the Observation Domain ID specified in the IPFIX Message Header. It MUST NOT contain new Template or Options Template Records, or any Data Sets. The Exporting Process SHOULD NOT send a Template Withdrawal Message until sufficient time has elapsed to allow receipt and processing of and Data Records described by the withdrawn Templates; see section 8.2 for more information on sequencing Template Withdrawals.

The end of a Transport Session implicitly withdraws all the Templates used within the Transport Session, and Templates must be resent during subsequent Transport Sessions between an Exporting Process and Collecting Process. All Templates for a given Observation Domain MAY also be withdrawn using an All Templates Withdrawal, which withdraws the special Template ID 2; this is shown in Figure U. All Options Templates for a given observation Domain MAY likewise be withdrawn using an All Options Templates Withdrawal, which withdraws the special Template ID 3. Each of these Withdrawals MUST appear in a Template Withdrawal Message with no other Withdrawals.

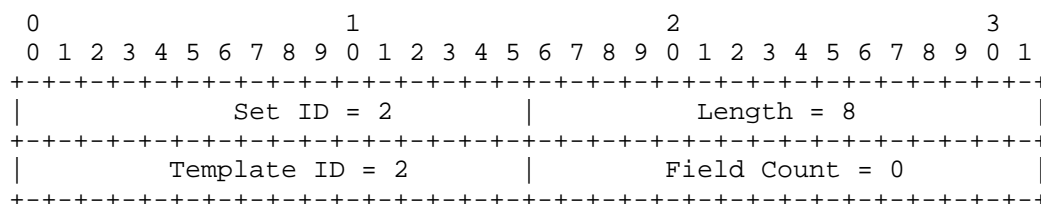


Figure U: All Templates Withdrawal Set Format

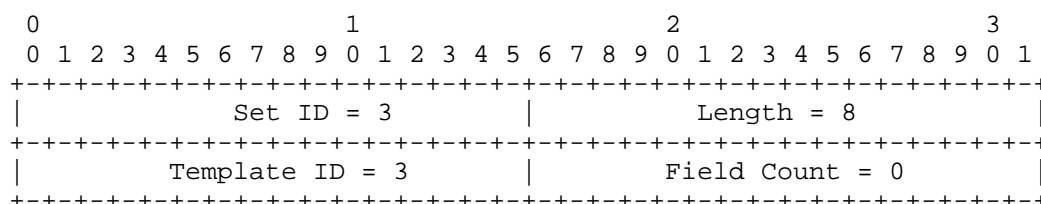


Figure V: All Options Templates Withdrawal Set Format

Template IDs MAY be reused for new Templates by sending a new Template Record or Options Template Record for a given Template ID after withdrawing the Template.

If a Collecting Process receives a new Template Record or Options Template Record for an already-allocated Template ID, without having received a withdrawal, it MUST ignore the new Template Record and discard the old Template Record for the allocated ID; it SHOULD log the error.

If a Collecting Process receives a Template Withdrawal for a Template or Options Template it does not presently have stored, it MUST ignore the Template Withdrawal and SHOULD log the error.

8.2 Sequencing Template Management Actions

Since there is no guarantee of the ordering of exported IPFIX Messages across SCTP Streams or over UDP, an Exporting Process MUST sequence all template management actions (i.e., Template Records defining new templates and Template Withdrawals withdrawing them) using the Export Time field in the IPFIX Message Header.

An Exporting Process MUST NOT export a Data Set described by a new Template in an IPFIX Message with an Export Time before the Export Time of the IPFIX Message containing that Template. If a new Template and a Data Set described by it appear in the same IPFIX Message, the

Template Set containing the Template MUST appear before the Data Set in the Message.

An Exporting Process MUST NOT export any Data Sets described by a withdrawn Template in IPFIX Messages with an Export Time after the Export Time of the IPFIX Message containing the Template Withdrawal withdrawing that Template.

Put another way, a Template only describes Records contained in IPFIX Messages with the same Export Time as the IPFIX Message containing Template Record, or a subsequent export time. Likewise, a Template Withdrawal is only in effect for IPFIX Messages with the same Export Time as the Template Withdrawal, or a subsequent Export Time.

Collecting Processes MAY implement a buffer to handle out-of-order Template management events.

8.3. Additional considerations for Template Management over SCTP

Template Sets and Options Template Sets MAY be sent on any SCTP stream. Data Sets sent on a given SCTP stream MAY be represented by Template Records exported on any SCTP stream.

Template Sets and Options Template Sets MUST be sent reliably and in order.

Template Withdrawal Messages MAY be sent on any SCTP stream. Template Withdrawal Messages MUST be sent reliably, using SCTP-ordered delivery. Template IDs MAY be reused by sending a Template Withdrawal Message and/or a new Template Record on a different SCTP stream than the stream on which the original Template was sent.

Additional Template Management considerations are given in [IPFIX-PER-SCTP-STREAM], which specifies an extension to explicitly link Templates with SCTP streams. In exchange for more restrictive rules on the assignment of Template Records to SCTP streams, this extension allows fast, reliable reuse of Template IDs and estimation of Data Record loss per Template.

8.4. Additional considerations for Template Management over UDP

Since UDP provides no method for reliable transmission of Templates, Exporting Processes using UDP as the Transport Protocol MUST periodically retransmit each active Template at regular intervals. The template retransmission interval MUST be configurable, as via the `templateRefreshTimeout` and `optionsTemplateRefreshTimeout` defined in [IPFIX-CONF]. Default settings for these values are deployment- and application-specific.

Before exporting any Data Records described by a given Template Record or Options Template Record, especially in the case of Template ID reuse as in section 8.1, the Exporting Process SHOULD send multiple copies of the Template Record in separate IPFIX Message, in order to help ensure the Collecting Process has received it.

In order to minimize resource requirements for templates which have expired at the Exporting Process without being withdrawn, or in cases when the Template Withdrawal Message was lost between the Exporting Process and the Collecting Process, the Collecting Process MAY associate a lifetime with each Template received in a UDP Transport Session. Templates not refreshed by the Exporting Process within the lifetime can then be discarded by the Collecting Process. The template lifetime at the Collecting Process MAY be exposed by a configuration parameter, or MAY be derived from observation of the interval of periodic Template retransmissions from the Exporting Process. In this latter case, the Template lifetime SHOULD default to at least 3 times the observed retransmission rate.

As template IDs are unique per UDP session and per Observation Domain, at any given time, the Collecting Process SHOULD maintain the following for all the current Template Records and Options Template Records: <IPFIX Device, Exporter source UDP port, Observation Domain ID, Template ID, Template Definition, Last Received>.

9. The Collecting Process's Side

This section describes the handling of the IPFIX Protocol at the Collecting Process common to all Transport Protocols. Additional considerations for SCTP and UDP are given in Sections 9.1 and 9.2 respectively. Template management at Collecting Processes is covered in Section 8.

The Collecting Process MUST listen for association requests / connections to start new Transport Sessions from the Exporting Process.

The Collecting Process MUST note the Information Element identifier of any Information Element that it does not understand and MAY discard that Information Element from the Flow Record.

The Collecting Process MUST accept padding in Data Records and Template Records. The padding size is the Set Length minus the size of the Set Header (4 octets for the Set ID and the Set Length), modulo the Record size deduced from the Template Record.

The IPFIX protocol has a Sequence Number field in the Export header that increases with the number of IPFIX Data Records in the IPFIX

Message. The Collecting Process MAY detect out-of-sequence, dropped, or duplicate IPFIX Messages using this the Sequence Number. If it supports this mechanism, the Collecting Process SHOULD log out-of-sequence IPFIX Messages, as these could indicate resource exhaustion at the Exporting Process or the Collecting Process, an Exporting Process reset, packet loss due to congestion between the Exporting Process and the Collecting Process, or message injection.

If the Collecting Process receives a malformed IPFIX Message, it MUST discard the IPFIX Message and SHOULD log the error. Note that non-zero Set padding does not constitute a malformed IPFIX Message.

9.1. Additional considerations for SCTP Collecting Processes

The Exporting Process requests a number of streams to use for export at association setup time. An Exporting Process MAY request and support more than one stream per SCTP association.

9.2. Additional considerations for UDP Collecting Processes

A Transport Session for IPFIX Messages transported over UDP is defined from the point of view of the Exporting Process, and roughly corresponds to the time during which a given Exporting Process sends IPFIX messages over UDP to a given Collecting Process. Since this is difficult to detect at the Collecting Process, the Collecting Process MAY expire all Transport Session state after no IPFIX Messages are received from a given Exporting Process during a configurable idle timeout.

The Collecting Process SHOULD accept Data Records without the associated Template Record (or other definitions) required to decode the Data Record. If the Template Records (or other definitions such as Common Properties) have not been received at the time Data Records are received, the Collecting Process SHOULD store the Data Records for a short period of time and decode them after the Template Records (or other definitions) are received. The short period of time MUST be lower than the lifetime of definitions associated with identifiers considered unique within the UDP session.

10. Transport Protocol

The IPFIX Protocol Specification has been designed to be transport protocol independent. Note that the Exporter can export to multiple Collecting Processes using independent transport protocols.

The IPFIX Message Header 16-bit Length field limits the length of an IPFIX Message to 65535 octets, including the header. A Collecting Process MUST be able to handle IPFIX Message lengths of up to 65535

octets.

10.1. Transport Compliance and Transport Usage

SCTP [RFC4960] using the PR-SCTP extension specified in [RFC3758] MUST be implemented by all compliant implementations. UDP [UDP] MAY also be implemented by compliant implementations. TCP [TCP] MAY also be implemented by compliant implementations.

SCTP SHOULD be used in deployments where Exporters and Collectors are communicating over links that are susceptible to congestion. PR-SCTP is capable of providing any required degree of reliability.

TCP MAY be used in deployments where Exporters and Collectors communicate over links that are susceptible to congestion, but SCTP is preferred due to its ability to limit back pressure on Exporters and its message versus stream orientation.

UDP MAY be used, although it is not a congestion-aware protocol. However, in this case the IPFIX traffic between Exporter and Collector MUST be separately contained or provisioned to minimize the risk of congestion-related loss.

10.2. SCTP

This section describes how IPFIX is transported over SCTP [RFC4960] using the PR-SCTP [RFC3758] extension.

10.2.1. Congestion Avoidance

The SCTP transport protocol provides the required level of congestion avoidance by design.

SCTP will detect congestion in the end-to-end path between the IPFIX Exporting Process and the IPFIX Collecting Process, and limit the transfer rate accordingly. When an IPFIX Exporting Process has records to export, but detects that transmission by SCTP is temporarily impossible, it can either wait until sending is possible again, or it can decide to drop the record. In the latter case, the dropped export data MUST be accounted for, so that the amount of dropped export data can be reported.

10.2.2. Reliability

The SCTP transport protocol is by default reliable, but has the capability to deliver messages with partial reliability [RFC3758].

Using reliable SCTP messages for the IPFIX export is not in itself a guarantee that all Data Records will be delivered. If there is congestion on the link from the Exporting Process to the Collecting

Process, or if a significant number of retransmissions are required, the send queues on the Exporting Process may fill up; the Exporting Process MAY either suspend, export, or discard the IPFIX Messages. If Data Records are discarded the IPFIX Sequence Numbers used for export MUST reflect the loss of data.

10.2.3. MTU

SCTP provides the required IPFIX Message fragmentation service based on path MTU discovery.

10.2.4. Association Establishment and Shutdown

The IPFIX Exporting Process SHOULD initiate an SCTP association with the IPFIX Collecting Process. By default, the Collecting Process listens for connections on SCTP port 4739. By default, the Collecting Process listens for secure connections on SCTP port 4740 (refer to the Security Considerations section). By default, the Exporting Process tries to connect to one of these ports. It MUST be possible to configure both the Exporting and Collecting Processes to use a different SCTP port.

The Exporting Process MAY establish more than one association (connection "bundle" in SCTP terminology) to the Collecting Process.

An Exporting Process MAY support more than one active association to different Collecting Processes (including the case of different Collecting Processes on the same host).

When an Exporting Process is shut down, it SHOULD shut down the SCTP association.

When a Collecting Process no longer wants to receive IPFIX Messages, it SHOULD shut down its end of the association. The Collecting Process SHOULD continue to receive and process IPFIX Messages until the Exporting Process has closed its end of the association.

When a Collecting Process detects that the SCTP association has been abnormally terminated, it MUST continue to listen for a new association establishment.

When an Exporting Process detects that the SCTP association to the Collecting Process is abnormally terminated, it SHOULD try to re-establish the association.

Association timeouts SHOULD be configurable.

10.2.5. Failover

If the Collecting Process does not acknowledge the attempt by the Exporting Process to establish an association, the Exporting Process should retry using the SCTP exponential backoff feature. The Exporter MAY log an alarm if the time to establish the association exceeds a specified threshold, configurable on the Exporter.

If Collecting Process failover is supported by the Exporting Process, a second SCTP association MAY be opened in advance.

10.2.6. Streams

An Exporting Process MAY request more than one SCTP stream per association. Each of these streams may be used for the transmission of IPFIX Messages containing Data Sets, Template Sets, and/or Options Template Sets.

Depending on the requirements of the application, the Exporting Process may send Data Sets with full or partial reliability, using ordered or out-of-order delivery, over any SCTP stream established during SCTP Association setup.

An IPFIX Exporting Process MAY use any PR-SCTP Service Definition as per Section 4 of the PR-SCTP [RFC3758] specification when using partial reliability to transmit IPFIX Messages containing only Data Sets.

However, Exporting Processes SHOULD mark such IPFIX Messages for retransmission for as long as resource or other constraints allow.

10.3. UDP

This section describes how IPFIX is transported over UDP [UDP].

10.3.1. Congestion Avoidance

UDP has no integral congestion-avoidance mechanism. Its use over congestion-sensitive network paths is therefore not recommended. UDP MAY be used in deployments where Exporters and Collectors always communicate over dedicated links that are not susceptible to congestion, i.e., links that are over-provisioned compared to the maximum export rate from the Exporters.

10.3.2. Reliability

UDP is not a reliable transport protocol, and cannot guarantee delivery of messages. IPFIX Messages sent from the Exporting Process

to the Collecting Process using UDP may therefore be lost. UDP MUST NOT be used unless the application can tolerate some loss of IPFIX Messages.

The Collecting Process SHOULD deduce the loss and reordering of IPFIX Data Records by looking at the discontinuities in the IPFIX Sequence Number. In the case of UDP, the IPFIX Sequence Number contains the total number of IPFIX Data Records sent for the UDP Transport Session prior to the receipt of this IPFIX Message, modulo 2^{32} . A Collector SHOULD detect out-of-sequence, dropped, or duplicate IPFIX Messages by tracking the Sequence Number. Templates sent from the Exporting Process to the Collecting Process using UDP as a transport MUST be re-sent at regular intervals, in case previous copies were lost.

Exporting Processes exporting IPFIX Messages via UDP MUST include a valid UDP checksum.

10.3.3. MTU

The maximum size of exported messages MUST be configured such that the total packet size does not exceed the path MTU. If the path MTU is unknown, a maximum packet size of 512 octets SHOULD be used.

10.3.4. Session Establishment and Shutdown

By default, the Collecting Process listens on the UDP port 4739. By default, the Collecting Process listens for secure connections on UDP port 4740 (refer to the "Security Considerations" section). By default, the Exporting Process tries to connect to one of these ports. It MUST be possible to configure both the Exporting and Collecting Processes to use a different UDP port.

As UDP is a connectionless protocol, there is no real session establishment or shutdown for IPFIX over UDP. An Exporting Process starts sending IPFIX Messages to a Collecting Process at one point in time, and stops sending them at another point in time. This leads to some complications in template management, which are outlined in Section 8.4 above.

10.3.5. Failover and Session Duplication

Because UDP is not a connection-oriented protocol, the Exporting Process is unable to determine from the transport protocol that the Collecting Process is no longer able to receive the IPFIX Messages. Therefore, it cannot invoke a failover mechanism. However, the Exporting Process MAY duplicate the IPFIX Message to several Collecting Processes.

10.4. TCP

The IPFIX Exporting Process initiates a TCP connection to the Collecting Process. By default, the Collecting Process listens for connections on TCP port 4739. By default, the Collecting Process listens for secure connections on TCP port 4740 (refer to the Security Considerations section). By default, the Exporting Process tries to connect to one of these ports. It **MUST** be possible to configure both the Exporting Process and the Collecting Process to use a different TCP port.

An Exporting Process **MAY** support more than one active connection to different Collecting Processes (including the case of different Collecting Processes on the same host).

The Exporter **MAY** log an alarm if the time to establish the connection exceeds a specified threshold, configurable on the Exporter.

10.4.1. Congestion Avoidance

TCP controls the rate at which data can be sent from the Exporting Process to the Collecting Process, using a mechanism that takes into account both congestion in the network and the capabilities of the receiver.

Therefore, an IPFIX Exporting Process may not be able to send IPFIX Messages at the rate that the Metering Process generates it, either because of congestion in the network or because the Collecting Process cannot handle IPFIX Messages fast enough. As long as congestion is transient, the Exporting Process can buffer IPFIX Messages for transmission. But such buffering is necessarily limited, both because of resource limitations and because of timeliness requirements, so ongoing and/or severe congestion may lead to a situation where the Exporting Process is blocked.

When an Exporting Process has Data Records to export but the transmission buffer is full, and it wants to avoid blocking, it can decide to drop some Data Records. The dropped Data Records **MUST** be accounted for, so that the number of lost records can later be exported as in Section 4.3.

When an Exporting Process finds that the rate at which records should be exported is consistently higher than the rate at which TCP sending permits, it **SHOULD** provide back pressure to the Metering Processes. The Metering Process could then adapt by temporarily reducing the amount of data it generates, for example, using sampling or aggregation.

10.4.2. Reliability

TCP ensures reliable delivery of data from the Exporting Process to the Collecting Process.

In the case of TCP, the IPFIX Sequence Number contains the total number of IPFIX Data Records sent from this TCP connection, from the current Observation Domain by the Exporting Process, prior to the receipt of this IPFIX Message, modulo 2^{32} .

10.4.3. MTU

As TCP offers a stream service instead of a datagram or sequential packet service, IPFIX Messages transported over TCP are instead separated using the Length field in the IPFIX Message Header. The Exporting Process can choose any valid length for exported IPFIX Messages, as TCP handles segmentation.

However, if an Exporting Process exports data from multiple Observation Domains, it should be careful to choose IPFIX Message lengths appropriately to minimize head-of-line blocking between different Observation Domains. Multiple TCP connections MAY be used to avoid head-of-line blocking between different Observation Domains.

10.4.4. Connection Establishment, Shutdown, and Restart

The IPFIX Exporting Process initiates a TCP connection to the Collecting Process. By default, the Collecting Process listens for connections on TCP port 4739. By default, the Collecting Process listens for secure connections on TCP port 4740 (refer to the Security Considerations section). By default, the Exporting Process tries to connect to one of these ports. It MUST be possible to configure both the Exporting Process and the Collecting Process to use a different TCP port.

An Exporting Process MAY support more than one active connection to different Collecting Processes (including the case of different Collecting Processes on the same host).

The Exporter MAY log an alarm if the time to establish the connection exceeds a specified threshold, configurable on the Exporter.

When an Exporting Process is shut down, it SHOULD shut down the TCP connection.

When a Collecting Process no longer wants to receive IPFIX Messages, it SHOULD close its end of the connection. The Collecting Process SHOULD continue to read IPFIX Messages until the Exporting Process

has closed its end.

When a Collecting Process detects that the TCP connection to the Exporting Process has terminated abnormally, it MUST continue to listen for a new connection.

When an Exporting Process detects that the TCP connection to the Collecting Process has terminated abnormally, it SHOULD try to re-establish the connection. Connection timeouts and retry schedules SHOULD be configurable. In the default configuration, an Exporting Process MUST NOT attempt to establish a connection more frequently than once per minute.

10.4.5. Failover

If the Collecting Process does not acknowledge the attempt by the Exporting Process to establish a connection, it will retry using the TCP exponential backoff feature.

If Collecting Process failover is supported by the Exporting Process, a second TCP connection MAY be opened in advance.

11. Security Considerations

The security considerations for the IPFIX protocol have been derived from an analysis of potential security threats, as discussed in the "Security Considerations" section of IPFIX requirements [RFC3917]. The requirements for IPFIX security are as follows:

1. IPFIX must provide a mechanism to ensure the confidentiality of IPFIX data transferred from an Exporting Process to a Collecting Process, in order to prevent disclosure of Flow Records transported via IPFIX.
2. IPFIX must provide a mechanism to ensure the integrity of IPFIX data transferred from an Exporting Process to a Collecting Process, in order to prevent the injection of incorrect data or control information (e.g., Templates) into an IPFIX Message stream.
3. IPFIX must provide a mechanism to authenticate IPFIX Collecting and Exporting Processes, to prevent the collection of data from an unauthorized Exporting Process or the export of data to an unauthorized Collecting Process.

Because IPFIX can be used to collect information for network forensics and billing purposes, attacks designed to confuse, disable, or take information from an IPFIX collection system may be seen as a

prime objective during a sophisticated network attack.

An attacker in a position to inject false messages into an IPFIX Message stream can either affect the application using IPFIX (by falsifying data), or the IPFIX Collecting Process itself (by modifying or revoking Templates, or changing options); for this reason, IPFIX Message integrity is important.

The IPFIX Messages themselves may also contain information of value to an attacker, including information about the configuration of the network as well as end-user traffic and payload data, so care must be taken to confine their visibility to authorized users. When an Information Element containing end-user payload information is exported, it SHOULD be transmitted to the Collecting Process using a means that secures its contents against eavesdropping. Suitable mechanisms include the use of either a direct point-to-point connection or the use of an encryption mechanism. It is the responsibility of the Collecting Process to provide a satisfactory degree of security for this collected data, including, if necessary, anonymization of any reported data.

11.1. Applicability of TLS and DTLS

Transport Layer Security (TLS) [RFC5246] and Datagram Transport Layer Security (DTLS) [RFC4347] were designed to provide the confidentiality, integrity, and authentication assurances required by the IPFIX protocol, without the need for pre-shared keys.

With the mandatory SCTP transport protocol for IPFIX, DTLS [RFC4347] MUST be implemented. If UDP is selected as the IPFIX transport protocol, DTLS [RFC4347] MUST be implemented. If TCP is selected as the IPFIX transport protocol, TLS [RFC5246] MUST be implemented.

Note that DTLS is selected as the security mechanism for SCTP. Though TLS bindings to SCTP are defined in [RFC3436], they require all communication to be over reliable, bidirectional streams, and require one TLS connection per stream. This arrangement is not compatible with the rationale behind the choice of SCTP as an IPFIX transport protocol.

Note that using DTLS [RFC4347] has a vulnerability, i.e., a true man in the middle may attempt to take data out of an association and fool the sender into thinking that the data was actually received by the peer. In generic TLS for SCTP (and/or TCP), this is not possible. This means that the removal of a message may become hidden from the sender or receiver. Another vulnerability of using SCTP with DTLS is that someone could inject SCTP control information to shut down the SCTP association, effectively generating a loss of IPFIX Messages if

those are buffered outside of the SCTP association. Techniques such as [RFC6083] could be used to overcome these vulnerabilities.

When using DTLS over SCTP, the Exporting Process MUST ensure that each IPFIX Message is sent over the same SCTP stream that would be used when sending the same IPFIX Message directly over SCTP. Note that DTLS may send its own control messages on stream 0 with full reliability; however, this will not interfere with the processing of stream 0 IPFIX Messages at the Collecting Process, because DTLS consumes its own control messages before passing IPFIX Messages up to the application layer.

When using DTLS over SCTP or UDP, the Heartbeat Extension [RFC6520] SHOULD be used, especially on long-lived Transport Sessions, to ensure that the association remains active.

11.2. Usage

The IPFIX Exporting Process initiates the communication to the IPFIX Collecting Process, and acts as a TLS or DTLS client according to [RFC5246] and [RFC4347], while the IPFIX Collecting Process acts as a TLS or DTLS server. The DTLS client opens a secure connection on the SCTP port 4740 of the DTLS server if SCTP is selected as the transport protocol. The TLS client opens a secure connection on the TCP port 4740 of the TLS server if TCP is selected as the transport protocol. The DTLS client opens a secure connection on the UDP port 4740 of the DTLS server if UDP is selected as the transport protocol.

11.3. Authentication

IPFIX Exporting Processes and IPFIX Collecting Processes are identified by the fully qualified domain name of the interface on which IPFIX Messages are sent or received, for purposes of X.509 client and server certificates as in [RFC5280].

To prevent man-in-the-middle attacks from impostor Exporting or Collecting Processes, the acceptance of data from an unauthorized Exporting Process, or the export of data to an unauthorized Collecting Process, strong mutual authentication via asymmetric keys MUST be used for both TLS and DTLS. Each of the IPFIX Exporting and Collecting Processes MUST verify the identity of its peer against its authorized certificates, and MUST verify that the peer's certificate matches its fully qualified domain name, or, in the case of SCTP, the fully qualified domain name of one of its endpoints.

The fully qualified domain name used to identify an IPFIX Collecting Process or Exporting Process may be stored either in a subjectAltName extension of type dNSName, or in the most specific Common Name field of the Subject field of the X.509 certificate. If both are present, the subjectAltName extension is given preference.

Internationalized domain names (IDN) in either the subjectAltName extension of type dNSName or the most specific Common Name field of the Subject field of the X.509 certificate MUST be encoded using Punycode [RFC3492] as described in [RFC5891], "Conversion Operations".

11.4. Protection against DoS Attacks

An attacker may mount a denial-of-service (DoS) attack against an IPFIX collection system either directly, by sending large amounts of traffic to a Collecting Process, or indirectly, by generating large amounts of traffic to be measured by a Metering Process.

Direct denial-of-service attacks can also involve state exhaustion, whether at the transport layer (e.g., by creating a large number of pending connections), or within the IPFIX Collecting Process itself (e.g., by sending Flow Records pending Template or scope information, a large amount of Options Template Records, etc.).

SCTP mandates a cookie-exchange mechanism designed to defend against SCTP state exhaustion denial-of-service attacks. Similarly, TCP provides the "SYN cookie" mechanism to mitigate state exhaustion; SYN cookies SHOULD be used by any Collecting Process accepting TCP connections. DTLS also provides cookie exchange to protect against DTLS server state exhaustion.

The reader should note that there is no way to prevent fake IPFIX Message processing (and state creation) for UDP & SCTP communication.

The use of TLS and DTLS can obviously prevent the creation of fake states, but they are themselves prone to state exhaustion attacks. Therefore, Collector rate limiting SHOULD be used to protect TLS & DTLS (like limiting the number of new TLS or DTLS session per second to a sensible number).

IPFIX state exhaustion attacks can be mitigated by limiting the rate at which new connections or associations will be opened by the Collecting Process, the rate at which IPFIX Messages will be accepted by the Collecting Process, and adaptively limiting the amount of state kept, particularly records waiting on Templates. These rate and state limits MAY be provided by a Collecting Process; if provided, the limits SHOULD be user configurable.

Additionally, an IPFIX Collecting Process can eliminate the risk of state exhaustion attacks from untrusted nodes by requiring TLS or DTLS mutual authentication, causing the Collecting Process to accept IPFIX Messages only from trusted sources.

With respect to indirect denial of service, the behavior of IPFIX under overload conditions depends on the transport protocol in use. For IPFIX over TCP, TCP congestion control would cause the flow of IPFIX Messages to back off and eventually stall, blinding the IPFIX system. SCTP improves upon this situation somewhat, as some IPFIX Messages would continue to be received by the Collecting Process due to the avoidance of head-of-line blocking by SCTP's multiple streams and partial reliability features, possibly affording some visibility of the attack. The situation is similar with UDP, as some datagrams may continue to be received at the Collecting Process, effectively applying sampling to the IPFIX Message stream, implying that some forensics may be left.

To minimize IPFIX Message loss under overload conditions, some mechanism for service differentiation could be used to prioritize IPFIX traffic over other traffic on the same link. Alternatively, IPFIX Messages can be transported over a dedicated network. In this case, care must be taken to ensure that the dedicated network can handle the expected peak IPFIX Message traffic.

11.5. When DTLS or TLS Is Not an Option

The use of DTLS or TLS might not be possible in some cases due to performance issues or other operational concerns.

Without TLS or DTLS mutual authentication, IPFIX Exporting Processes and Collecting Processes can fall back on using IP source addresses to authenticate their peers. A policy of allocating Exporting Process and Collecting Process IP addresses from specified address ranges, and using ingress filtering to prevent spoofing, can improve the usefulness of this approach. Again, completely segregating IPFIX traffic on a dedicated network, where possible, can improve security even further. In any case, the use of open Collecting Processes (those that will accept IPFIX Messages from any Exporting Process regardless of IP address or identity) is discouraged.

Modern TCP and SCTP implementations are resistant to blind insertion attacks (see [RFC1948], [RFC4960]); however, UDP offers no such protection. For this reason, IPFIX Message traffic transported via UDP and not secured via DTLS SHOULD be protected via segregation to a dedicated network.

11.6. Logging an IPFIX Attack

IPFIX Collecting Processes MUST detect potential IPFIX Message insertion or loss conditions by tracking the IPFIX Sequence Number, and SHOULD provide a logging mechanism for reporting out-of-sequence messages. Note that an attacker may be able to exploit the handling of out-of-sequence messages at the Collecting Process, so care should be taken in handling these conditions. For example, a Collecting Process that simply resets the expected Sequence Number upon receipt of a later Sequence Number could be temporarily blinded by deliberate injection of later Sequence Numbers.

IPFIX Exporting and Collecting Processes SHOULD log any connection attempt that fails due to authentication failure, whether due to being presented an unauthorized or mismatched certificate during TLS or DTLS mutual authentication, or due to a connection attempt from an unauthorized IP address when TLS or DTLS is not in use.

IPFIX Exporting and Collecting Processes SHOULD detect and log any SCTP association reset or TCP connection reset.

11.7. Securing the Collector

The security of the Collector and its implementation is important to achieve overall security. However, it is outside the scope of this document.

12. IANA Considerations

IPFIX Messages use two fields with assigned values. These are the IPFIX Version Number, indicating which version of the IPFIX Protocol was used to export an IPFIX Message, and the IPFIX Set ID, indicating the type for each set of information within an IPFIX Message.

The IPFIX Version Number value of 10 is reserved for the IPFIX protocol specified in this document. Set ID values of 0 and 1 are not used for historical reasons [RFC3954]. The Set ID value of 2 is reserved for the Template Set. The Set ID value of 3 is reserved for the Options Template Set. All other Set ID values from 4 to 255 are reserved for future use. Set ID values above 255 are used for Data Sets.

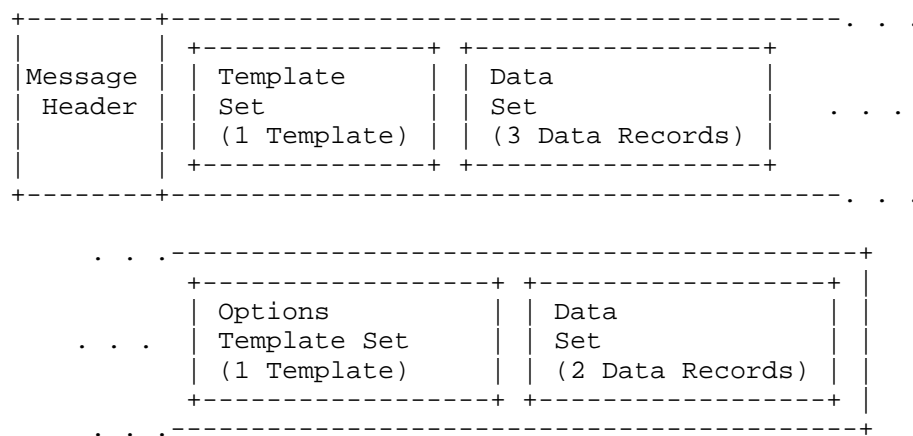
New assignments in either IPFIX Version Number or IPFIX Set ID assignments require a Standards Action [RFC5226], i.e., they are to be made via Standards Track RFCs approved by the IESG.

Appendix A. IPFIX Encoding Examples

This appendix, which is a not a normative reference, contains IPFIX encoding examples.

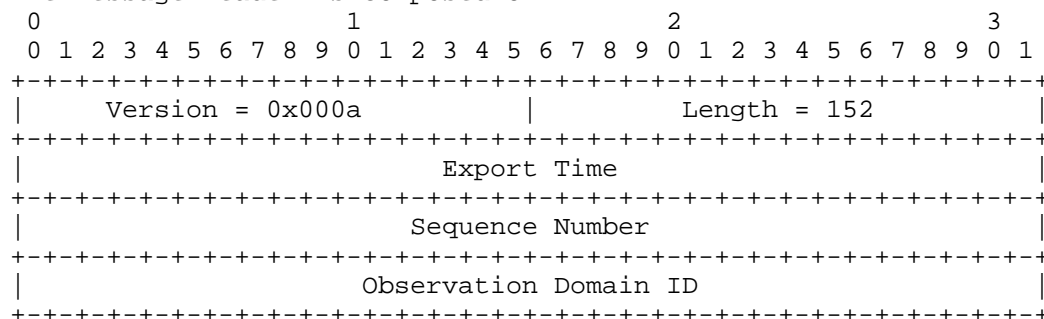
Let's consider the example of an IPFIX Message composed of a Template Set, a Data Set (which contains three Data Records), an Options Template Set and a Data Set (which contains 2 Data Records related to the previous Options Template Record).

IPFIX Message:



A.1. Message Header Example

The Message Header is composed of:



A.2. Template Set Examples

A.2.1. Template Set Using IETF-Specified Information Elements

We want to report the following Information Elements:

- The IPv4 source IP address: sourceIPv4Address in [RFC5102bis], with a length of 4 octets
- The IPv4 destination IP address: destinationIPv4Address in [RFC5102bis], with a length of 4 octets
- The next-hop IP address (IPv4): ipNextHopIPv4Address in [RFC5102bis], with a length of 4 octets
- The number of packets of the Flow: packetDeltaCount in [RFC5102bis], with a length of 4 octets
- The number of octets of the Flow: octetDeltaCount in [RFC5102bis], with a length of 4 octets

Therefore, the Template Set will be composed of the following:

0	1																2																3																
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9										
Set ID = 2																	Length = 28 octets																																
Template ID 256																	Field Count = 5																																
0	sourceIPv4Address = 8																Field Length = 4																																
0	destinationIPv4Address = 12																Field Length = 4																																
0	ipNextHopIPv4Address = 15																Field Length = 4																																
0	packetDeltaCount = 2																Field Length = 4																																
0	octetDeltaCount = 1																Field Length = 4																																

A.2.2. Template Set Using Enterprise-Specific Information Elements

We want to report the following Information Elements:

- The IPv4 source IP address: sourceIPv4Address in [RFC5102bis], with a length of 4 octets

- The IPv4 destination IP address: destinationIPv4Address in [RFC5102bis], with a length of 4 octets
- An enterprise-specific Information Element representing proprietary information, with a type of 15 and a length of 4
- The number of packets of the Flow: packetDeltaCount in [RFC5102bis], with a length of 4 octets
- The number of octets of the Flow: octetDeltaCount in [RFC5102bis], with a length of 4 octets

Therefore, the Template Set will be composed of the following:

0																1																2																3															
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9																								
Set ID = 2																Length = 32 octets																																															
Template ID 257																Field Count = 5																																															
0 sourceIPv4Address = 8																Field Length = 4																																															
0 destinationIPv4Address = 12																Field Length = 4																																															
1 Information Element Id. = 15																Field Length = 4																																															
Enterprise number																																																															
0 packetDeltaCount = 2																Field Length = 4																																															
0 octetDeltaCount = 1																Field Length = 4																																															

A.3. Data Set Example

In this example, we report the following three Flow Records:

Src IP addr.	Dst IP addr.	Next Hop addr.	Packet Number	Octets Number
192.0.2.12	192.0.2.254	192.0.2.1	5009	5344385
192.0.2.27	192.0.2.23	192.0.2.2	748	388934
192.0.2.56	192.0.2.65	192.0.2.3	5	6534

0

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

1

2

3

Set ID = 256

Length = 64

192.0.2.12

192.0.2.254

192.0.2.1

5009

5344385

192.0.2.27

192.0.2.23

192.0.2.2

748

388934

192.0.2.56

192.0.2.65

192.0.2.3

5

6534

Note that padding is not necessary in this example.

A.4. Options Template Set Examples

A.4.1. Options Template Set Using IETF-Specified Information Elements

Per line card (the router being composed of two line cards), we want to report the following Information Elements:

- Total number of IPFIX Messages: exportedMessageTotalCount [RFC5102bis], with a length of 2 octets
- Total number of exported Flows: exportedFlowRecordTotalCount [RFC5102bis], with a length of 2 octets

The line card, which is represented by the lineCardId Information Element [RFC5102bis], is used as the Scope Field.

Therefore, the Options Template Set will be:

0										1										2										3									
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1								
Set ID = 3										Length = 24																													
Template ID 258										Field Count = 3																													
Scope Field Count = 1										lineCardId = 141																													
Scope 1 Field Length = 4										exportedMessageTotalCount=41																													
Field Length = 2										exportedFlowRecordTotalCo.=42																													
Field Length = 2										Padding																													

A.4.2. Options Template Set Using Enterprise-Specific Information Elements

Per line card (the router being composed of two line cards), we want to report the following Information Elements:

- Total number of IPFIX Messages: exportedMessageTotalCount [RFC5102bis], with a length of 2 octets
- An enterprise-specific number of exported Flows, with a type of 42 and a length of 4 octets

The line card, which is represented by the lineCardId Information

Element [RFC5102bis], is used as the Scope Field.

The format of the Options Template Set is as follows:

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Set ID = 3           |           Length = 28           |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Template ID 259      |           Field Count = 3       |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Scope Field Count = 1 | 0 |           lineCardId = 141 |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Scope 1 Field Length = 4 | 0 | exportedFlowRecordTotalCo.=41 |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Field Length = 2      | 1 | Information Element Id. = 42 |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Field Length = 4      |           Enterprise number    ...
+-----+-----+-----+-----+-----+-----+-----+-----+
...           Enterprise number    |           Padding             |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

A.4.3. Options Template Set Using an Enterprise-Specific Scope

In this example, we want to export the same information as in the example in Section A.4.1:

- Total number of IPFIX Messages: exportedMessageTotalCount [RFC5102bis], with a length of 2 octets
- Total number of exported Flows: exportedFlowRecordTotalCount [RFC5102bis], with a length of 2 octets

But this time, the information pertains to a proprietary scope, identified by enterprise-specific Information Element number 123.

The format of the Options Template Set is now as follows:

```

      0               1               2               3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Set ID = 3           |           Length = 28           |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Template ID 260       |           Field Count = 3       |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Scope Field Count = 1   | 1 | Scope 1 Infor. El. Id. = 123 |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Scope 1 Field Length = 4 |           Enterprise Number   ...
+-----+-----+-----+-----+-----+-----+-----+-----+
...           Enterprise Number     | 0 | exportedMessageTotalCount=41 |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Field Length = 2       | 0 | exportedFlowRecordTotalCo.=42 |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Field Length = 2       |           Padding              |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

A.4.4. Data Set Using an Enterprise-Specific Scope

In this example, we report the following two Data Records:

Enterprise field 123	IPFIX Message	Exported Flow Records
1	345	10201
2	690	20402


```

      0               1               2               3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Set ID = 260           |           Length = 20           |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                   |           1                   |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           345                     |           10201                 |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                   |           2                   |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           690                     |           20402                 |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

A.5. Variable-Length Information Element Examples

A.5.1. Example of Variable-Length Information Element with Length Inferior to 255 Octets

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|           5           |           5 octet Information Element       |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     |
+-----+-----+-----+-----+-----+-----+-----+

```

A.5.2. Example of Variable-Length Information Element with 3 Octet Length Encoding

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|           255           |           1000           |           IE ...   |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           1000 octet Information Element           |
+-----+-----+-----+-----+-----+-----+-----+-----+
:                                     ...                                     :
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     ... IE           |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

References

Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3436] Jungmaier, A., Rescorla, E., and M. Tuexen, "Transport Layer Security over Stream Control Transmission Protocol", RFC 3436, December 2002.
- [RFC3492] Costello, A., "Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA)", RFC 3492, March 2003.
- [RFC3758] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", RFC 3758, May 2004.

- [RFC4347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security", RFC 4347, April 2006.
- [RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, September 2007.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.
- [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, April 2008.
- [RFC5905] Mills, D., Delaware, U., Martin, J., Burbank, J. and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010
- [RFC5891] J. Klensin, "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, August 2010.
- [RFC6520] Seggelmann, R., Tuexen, M., and Williams, M., "Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension", RFC 6520, February 2012.
- [TCP] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.
- [UDP] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.
- [RFC5102bis] Quittek, J., Bryant S., Claise, B., Aitken, P., and J. Meyer, "Information Model for IP Flow Information Export", draft-claise-ipfix-information-model-rfc5102bis-01.txt, Work in Progress, October 2011.

Informative References

- [PEN] IANA Private Enterprise Numbers registry
<http://www.iana.org/assignments/enterprise-numbers>.

- [RFC1948] Bellovin, S., "Defending Against Sequence Number Attacks", RFC 1948, May 1996.
- [RFC2579] McCloghrie, K., Perkins, D., and J. Schoenwaelder, "Textual Conventions for SMIV2", STD 58, RFC 2579, April 1999.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC3917] Quittek, J., Zseby, T., Claise, B., and S. Zander, "Requirements for IP Flow Information Export (IPFIX)", RFC 3917, October 2004.
- [RFC3954] Claise, B., Ed., "Cisco Systems NetFlow Services Export Version 9", RFC 3954, October 2004.
- [RFC5101] Claise, B., Ed., "Bidirectional Flow Export Using IP Flow Information Export (IPFIX)", RFC 5103, January 2008.
- [RFC5103] Trammell, B., and E. Boschi, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information", RFC 5101, January 2008.
- [RFC5153] Boschi, E., Mark, L., Quittek J., and P. Aitken, "IP Flow Information Export (IPFIX) Implementation Guidelines", RFC5153, April 2008
- [RFC5470] Sadasivan, G., Brownlee, N., Claise, B., and J. Quittek, "Architecture for IP Flow Information Export", RFC5470, March 2009.
- [RFC5472] Zseby, T., Boschi, E., Brownlee, N., and B. Claise, "IP Flow Information Export (IPFIX) Applicability", RFC5472, March 2009.
- [RFC5471] Schmoll, C., Aitken, P., and B. Claise, "Guidelines for IP Flow Information Export (IPFIX) Testing", RFC5471, March 2009
- [RFC5473] Boschi, E., Mark, L., and B. Claise, "Reducing Redundancy in IP Flow Information Export (IPFIX) and Packet Sampling (PSAMP) Reports", RFC5473, March 2009
- [RFC5610] Boschi, E., Trammell, B., Mark, L., and T. Zseby,

"Exporting Type Information for IP Flow Information Export (IPFIX) Information Elements", July 2009.

- [RFC6083] Tuexen, M., Seggelman, R. and E. Rescola, "Datagram Transport Layer Security (DTLS) for Stream Control Transmission Protocol (SCTP)", RFC6083, January 2011.
- [RFC6313] Claise, B., Dhandapani, G., Aitken, P, and S. Yates, "Export of Structured Data in IP Flow Information Export (IPFIX)", RFC6313, July 2011.
- [RFC6183] Kobayashi, A., Claise, B., Muenz, G, and K. Ishibashi, "IP Flow Information Export (IPFIX) Mediation: Framework", RFC6183, April 2011.
- [POSIX.1] IEEE 1003.1-2008 - IEEE Standard for Information Technology - Portable Operating System Interface, IEEE, 2008.
- [IEEE.754.1985] Institute of Electrical and Electronics Engineers, "Standard for Binary Floating-Point Arithmetic", IEEE Standard 754, August 1985.
- [IPFIX-CONF] Muenz, G., Claise, B., and P. Aitken, "Configuration Data Model for IPFIX and PSAMP", draft-ietf-ipfix-configuration-model-10, Work in Progress, July 2011.
- [IPFIX-PER-SCTP-STREAM] Claise, B., Aitekn, P., Johnson, A. and G. Muenz, "IPFIX Export per SCTP Stream", draft-ietf-ipfix-export-per-sctp-stream-08, Work in Progress, May 2010.
- [IPFIX-MED-PROTO] Claise, B., Kobayashi, A., and B. Trammell, "Specification of the Protocol for IPFIX Mediations", draft-claise-ipfix-mediation-protocol-04, Work in Progress, July 2011.
- [RFC5815bis] Dietz, T., Kobayashi, A., Claise, B., and G. Muenz, "Definitions of Managed Objects for IP Flow Information Export", draft-dkcm-ipfix-rfc5815bis-00.txt, Work in Progress, October 2011.

Acknowledgments

We would like to thank the following persons: Ganesh Sadasivan for his significant contribution during the initial phases of the protocol specification; Juergen Quittek for the coordination job within IPFIX and PSAMP; Nevil Brownlee, Dave Plonka, Paul Aitken, and

Andrew Johnson for the thorough reviews; Randall Stewart and Peter Lei for their SCTP expertise and contributions; Martin Djernaes for the first essay on the SCTP section; Michael Behringer and Eric Vyncke for their advice and knowledge in security; Michael Tuexen for his help regarding the DTLS section; Elisa Boschi for her contribution regarding the improvement of SCTP sections; Mark Fullmer, Sebastian Zander, Jeff Meyer, Maurizio Molina, Carter Bullard, Tal Givoly, Lutz Mark, David Moore, Robert Lowe, Paul Calato, Andrew Feren, Gerhard Muenz, and many more, for the technical reviews and feedback.

Authors' Addresses

Benoit Claise (Ed.)
Cisco Systems
De Kleetlaan 6a b1
1831 Diegem
Belgium

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

Brian Trammell (Ed.)
Swiss Federal Institute of Technology Zurich
Gloriastrasse 35
8092 Zurich
Switzerland

Phone: +41 44 632 70 13
EMail: trammell@tik.ee.ethz.ch

Stewart Bryant
Cisco Systems, Inc.
250, Longwater,
Green Park,
Reading, RG2 6GB,
United Kingdom

Phone: +44 (0)20 8824-8828
EMail: stbryant@cisco.com

Simon Leinen
SWITCH
Werdstrasse 2
P.O. Box
8021 Zurich
Switzerland

Phone: +41 44 268 1536
EMail: simon.leinen@switch.ch

Thomas Dietz
NEC Europe Ltd.
NEC Laboratories Europe
Network Research Division
Kurfuersten-Anlage 36
69115 Heidelberg
Germany

Phone: +49 6221 4342-128
EMail: Thomas.Dietz@nw.neclab.eu

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 11, 2013

T. Dietz, Ed.
NEC Europe Ltd.
B. Claise
Cisco Systems, Inc.
J. Quittek
NEC Europe Ltd.
July 10, 2012

Definitions of Managed Objects for Packet Sampling
<draft-ietf-ipfix-psamp-mib-06.txt>

Abstract

This memo defines a portion of the Management Information Base (MIB) for use with network management protocols in the Internet community. In particular, it describes extensions to the IPFIX SELECTOR MIB module. For IPFIX implementations that use packet Sampling (PSAMP) techniques, this memo defines the PSAMP MIB module containing managed objects for providing information on applied packet selection functions and their parameters.

Status of this Memo

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

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

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

This Internet-Draft will expire on January 11, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. The Internet-Standard Management Framework	3
2. Introduction	3
3. PSAMP Documents Overview	4
4. Related IPFIX Documents	4
5. Structure of the PSAMP MIB module	4
5.1. Textual Conventions	5
5.2. Packet Selection Functions	6
5.2.1. Systematic Count-based Sampling	6
5.2.2. Systematic Time-based Sampling	7
5.2.3. Random n-out-of-N Sampling	7
5.2.4. Uniform Probabilistic Sampling	7
5.2.5. Property Match Filtering	8
5.2.6. Hash-based Filtering	8
6. Definitions	8
7. Security Considerations	25
8. IANA Considerations	25
9. Acknowledgment	26
10. References	26
10.1. Normative References	26
10.2. Informative References	27
Authors' Addresses	27

1. The Internet-Standard Management Framework

For a detailed overview of the documents that describe the current Internet-Standard Management Framework, please refer to section 7 of RFC 3410 [RFC3410].

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. MIB objects are generally accessed through the Simple Network Management Protocol (SNMP). Objects in the MIB are defined using the mechanisms defined in the Structure of Management Information (SMI). This memo specifies a MIB module that is compliant to the SMIV2, which is described in STD 58, RFC 2578 [RFC2578], STD 58, RFC 2579 [RFC2579] and STD 58, RFC 2580 [RFC2580].

2. Introduction

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

This document is a product of the IP Flow Information eXport (IPFIX) working group. Work on this document was started in the Packet Sampling (PSAMP) Working Group (WG) and moved to the IPFIX WG when the PSAMP WG was concluded.

Its purpose is to define managed objects for monitoring PSAMP Devices performing packet selection by Sampling and Filtering as described in [RFC5475].

It is assumed that packet Sampling is performed according to the framework defined in [RFC5474].

Managed objects in the PSAMP MIB module are defined as an extension of the IPFIX MIB and IPFIX SELECTOR MIB modules [RFC6615]. Since the IPFIX MIB module is only for monitoring the same holds true for the PSAMP MIB module defined in this document. The definition of objects is in line with the PSAMP information model [RFC5477].

Section 3 gives an overview of the PSAMP documents, while section 4 refers to the related IPFIX documents. Section 5 describes the structure of the PSAMP MIB module and section 6 contains the formal definition. Security issues are discussed in section 7.

3. PSAMP Documents Overview

[RFC5474]: "A Framework for Packet Selection and Reporting" 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.

[RFC5475]: "Sampling and Filtering Techniques for IP Packet Selection" describes the set of packet selection techniques supported by PSAMP.

[RFC5476]: "Packet Sampling (PSAMP) Protocol Specifications" specifies the export of packet information from a PSAMP Exporting Process to a PSAMP Collecting Process.

[RFC5477]: "Information Model for Packet Sampling Exports" defines an information and data model for PSAMP.

This document: "Definitions of Managed Objects for Packet Sampling" describes the PSAMP Management Information Base.

4. Related IPFIX Documents

The IPFIX protocol provides network administrators with access to IP Flow information.

[RFC5101]: The protocol document "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information" specifies how IPFIX Data Records and Templates are carried via a congestion-aware transport protocol from IPFIX Exporting Processes to IPFIX Collecting Processes. It also specifies the data types used in the PSAMP MIB module and their encoding.

[RFC6615]: The IPFIX MIB "Definitions of Managed Objects for IP Flow Information Export" is the basis for this document because it extends the IPFIX SELECTOR MIB module defined there.

5. Structure of the PSAMP MIB module

The IPFIX MIB module defined in [RFC6615] has the concept of a packet Selection Process containing a set of Selector function instances. Selection Processes and functions are referenced in the `ipfixSelectionProcessTable` of the IPFIX MIB module. The `ipfixSelectionProcessTable` identifies an instance of a Selector function by an OID. The OID points to an object that describes the Selector function. For simple Selector functions without parameters,

the OID refers to an object which contains only one additional object indicating the current availability of the function. For functions which have one or more parameters the object has a subtree that in addition to an availability object contains a table with a conceptual column for each parameter. Entries (conceptual rows) in this table represent different combinations of parameter values for instances of the Selector function.

The object `ipfixSelectorFunctions` in the IPFIX SELECTOR MIB module serves as the root for objects that describe instances of packet Selector functions. The IPFIX SELECTOR MIB module is a very small module which is defined in [RFC6615]. The top level OIDs of the parameter trees located beneath `ipfixSelectorFunctions` are maintained by IANA. In the IPFIX SELECTOR MIB module as defined by [RFC6615] the object `ipfixSelectorFunctions` contains just a single trivial packet Selector function called `ipfixFuncSelectAll` that selects every packet and has no parameter:

```
ipfixSelectorMIB
+-- ipfixSelectorObjects(1)
    +-- ipfixSelectorFunctions(1)
        +-- ipfixFuncSelectAll(1)
            +-- ipfixFuncSelectAllAvail(1)
```

The PSAMP MIB module defined in this document registers additional toplevel OIDs for the parameter subtrees of its Selector functions in the IPFIX-SELECTOR-MIB Function subregistry according to the procedures defined in [RFC6615]. It introduces six new subtrees beneath `ipfixSelectorFunctions`. Each of them describes a packet Selector function with one or more parameters. Naming and ordering of objects is fully in line with the guidelines given in section 6.1 of [RFC6615]. All functions and their parameters are already listed in the overview of functions given by the table in section 8.2.1 of [RFC5477].

5.1. Textual Conventions

The PSAMP MIB module imports two textual conventions which define data types used in this MIB module from other MIB modules. The `Unsigned64TC` data type is imported from the APPLICATION MIB module [RFC2564] and the `Float64TC` data type is imported from the FLOAT-TC-MIB module [RFC6340]. Those data types are defined according to [RFC5101]. Those data types are not an integral part of [RFC2578] but are needed to define objects in this MIB module that conform to the Information Elements defined for those objects in [RFC5477].

The `Unsigned64TC` textual convention describes an unsigned integer of 64 bits. It is imported from the APPLICATION MIB module. The

Float64TC textual convention describes the format that is used for 64 bit floating point numbers.

5.2. Packet Selection Functions

In general, different packet Selector functions have different parameters. The PSAMP MIB module contains six objects with subtrees that provide information on parameters of function instances of different Selector functions. All objects are named and structured according to section 8.2.1 of [RFC5477]:

```
ipfixSelectorFunctions(1)
+-- psampSampCountBased(2)
+- -psampSampTimeBased(3)
+-- psampSampRandOutOfN(4)
+-- psampSampUniProb(5)
+-- psampFiltPropMatch(6)
+-- psampFiltHash(7)
```

Indexing of these functions in the PSAMP MIB module starts with index (2). The function ipfixFuncSelectAll with index (1) is already defined in the IPFIX SELECTOR MIB module as shown above.

The object tree for each of these functions is described below. Semantics of all functions and their parameters are described in detail in [RFC5475]. More information on the Selector Reports can also be found in section 6.5.2 of [RFC5476].

5.2.1. Systematic Count-based Sampling

The first Selector function is systematic count-based Sampling. Its availability is indicated by object psampSampCountBasedAvail. The function has two parameters: psampSampCountBasedInterval and psampSampCountBasedSpace. Different combination of values of these parameters for different instances of the Selector function are represented by different conceptual rows in table psampSampCountBasedParamSetEntry:

```
psampSampCountBased(2)
+-- psampSampCountBasedAvail(1)
+-- psampSampCountBasedParamSetTable(2)
    +-- psampSampCountBasedParamSetEntry(1) [psampSampCountBasedIndex]
        +-- psampSampCountBasedIndex(1)
        +-- psampSampCountBasedInterval(2)
        +-- psampSampCountBasedSpace(3)
```

5.2.2. Systematic Time-based Sampling

The second Selector function is systematic time-based Sampling. The structure of the sub-tree for this function is similar to the psampSampCountBased sub-tree. Parameters are psampSampTimeBasedInterval and psampSampTimeBasedSpace. They appear to be the same as for count based Sampling, but their data types are different because they indicate time values instead of numbers of packets:

```
psampSampTimeBased(3)
+-- psampSampTimeBasedAvail(1)
+-- psampSampTimeBasedParamSetTable(2)
    +-- psampSampTimeBasedParamSetEntry(1) [psampSampTimeBasedIndex]
        +-- psampSampTimeBasedIndex(1)
        +-- psampSampTimeBasedInterval(2)
        +-- psampSampTimeBasedSpace(3)
```

5.2.3. Random n-out-of-N Sampling

The third Selector function is random n-out-of-N Sampling. Parameters are psampSampRandOutOfNSize and psampSampRandOutOfNPopulation:

```
psampSampRandOutOfN(4)
+-- psampSampRandOutOfNAvail(1)
+-- psampSampRandOutOfNParamSetTable(2)
    +-- psampSampRandOutOfNParamSetEntry(1) [psampSampRandOutOfNIndex]
        +-- psampSampRandOutOfNIndex(1)
        +-- psampSampRandOutOfNSize(2)
        +-- psampSampRandOutOfNPopulation(3)
```

5.2.4. Uniform Probabilistic Sampling

The fourth Selector function is uniform probabilistic Sampling. It has just a single parameter called psampSampUniProbProbability:

```
psampSampUniProb(5)
+-- psampSampUniProbAvail(1)
+-- psampSampUniProbParamSetTable(2)
    +-- psampSampUniProbParamSetEntry(1) [psampSampUniProbIndex]
        +-- psampSampUniProbIndex(1)
        +-- psampSampUniProbProbability(2)
```

5.2.5. Property Match Filtering

The fifth Selector function is property match Filtering. For this Selector function there is a broad variety of possible parameters that could be used. But as stated in section 8.2.1 of [RFC5477] there are no agreed parameters specified and the sub-tree for this function only contains an object indicating the availability of this function. Parameters cannot be retrieved via the PSAMP MIB module:

```
psampFiltPropMatch(6)
+-- psampFiltPropMatchAvail(1)
```

5.2.6. Hash-based Filtering

The sixth Selector function is hash-based Filtering. The object `psampFiltHashFunction` is an enumeration that specifies the kind of hash function that is applied. These hash function have quite a number of parameters and the actual number may vary with the choice of the hash function applied. The common parameter set for all hash-based Filtering functions contains 7 parameters:

`psampFiltHashInitializerValue`, `psampFiltHashIpPayloadOffset`, `psampFiltHashIpPayloadSize`, `psampFiltHashSelectedRangeMin`, `psampFiltHashSelectedRangeMax`, `psampFiltHashOutputRangeMin`, and `psampFiltHashOutputRangeMax`.

```
psampFiltHash(7)
+-- psampFiltHashAvail(1)
+-- psampFiltHashCapabilities(2)
+-- psampFiltHashParamSetTable(3)
    +-- psampFiltHashParamSetEntry(1) [psampFiltHashIndex]
        +-- psampFiltHashIndex(1)
        +-- psampFiltHashFunction(2)
        +-- psampFiltHashInitializerValue(3)
        +-- psampFiltHashIpPayloadOffset(4)
        +-- psampFiltHashIpPayloadSize(5)
        +-- psampFiltHashSelectedRangeMin(6)
        +-- psampFiltHashSelectedRangeMax(7)
        +-- psampFiltHashOutputRangeMin(8)
        +-- psampFiltHashOutputRangeMax(9)
```

Further parameters depend on the applied hash function and are not specified within the PSAMP MIB module.

6. Definitions

```
PSAMP-MIB DEFINITIONS ::= BEGIN
```

IMPORTS

```
MODULE-IDENTITY, OBJECT-TYPE, Integer32, Unsigned32, mib-2
  FROM SNMPv2-SMI                      -- RFC2578
TruthValue
  FROM SNMPv2-TC                      -- RFC2579
MODULE-COMPLIANCE, OBJECT-GROUP
  FROM SNMPv2-CONF                    -- RFC2580
Unsigned64TC
  FROM APPLICATION-MIB                -- RFC2564
Float64TC
  FROM FLOAT-TC-MIB                   -- RFC6340
ipfixSelectorFunctions
  FROM IPFIX-SELECTOR-MIB;            -- RFC6615
```

psampMIB MODULE-IDENTITY

```
LAST-UPDATED "201207051200Z"          -- 05 July 2012
ORGANIZATION "IETF IPFIX Working Group"
CONTACT-INFO
  "WG charter:
   http://www.ietf.org/html.charters/ipfix-charter.html
```

Mailing Lists:

```
  General Discussion: ipfix@ietf.org
  To Subscribe: http://www1.ietf.org/mailman/listinfo/ipfix
  Archive:
http://www1.ietf.org/mail-archive/web/ipfix/current/index.html
```

Editor:

```
Thomas Dietz
NEC Europe Ltd.
NEC Laboratories Europe
Network Research Division
Kurfuersten-Anlage 36
69115 Heidelberg
Germany
Phone: +49 6221 4342-128
Email: Thomas.Dietz@neclab.eu
```

```
Benoit Claise
Cisco Systems, Inc.
De Kleetlaan 6a b1
Degem 1831
Belgium
Phone: +32 2 704 5622
Email: bclaise@cisco.com
```

```
Juergen Quittek
NEC Europe Ltd.
```

NEC Laboratories Europe
Network Research Division
Kurfuersten-Anlage 36
69115 Heidelberg
Germany
Phone: +49 6221 4342-115
Email: quittek@neclab.eu"

DESCRIPTION

"The PSAMP MIB defines managed objects for packet sampling and filtering.

These objects provide information about managed nodes supporting packet sampling, including packet sampling capabilities, configuration and statistics.

The PSAMP MIB module registers additional toplevel OIDs for the parameter subtrees of its Selector functions in the IPFIX-SELECTOR-MIB Function subregistry according to the procedures defined in RFC 6615.

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved. This version of this MIB module is part of RFC yyyy; see the RFC itself for full legal notices"

-- RFC Ed.: replace yyyy with actual RFC number & remove this notice
-- Revision history

REVISION "201207051200Z" -- 05 July 2012

DESCRIPTION

"Initial version, published as RFC yyyy."

-- RFC Ed.: replace yyyy with actual RFC number & remove this notice

::= { mib-2 xxx }

-- RFC Ed.: replace xxx which is to be assigned by IANA & remove
-- this notice.

-- Top level structure of the MIB

psampObjects OBJECT IDENTIFIER ::= { psampMIB 1 }
psampConformance OBJECT IDENTIFIER ::= { psampMIB 2 }

-- Packet selection sampling methods group of objects

--* Method 1: Systematic count-based Sampling

```

-- Reference: RFC5475, Section 5.1, RFC5476 Section 6.5.2.1 and
--            RFC5477, Section 8.2
psampSampCountBased OBJECT IDENTIFIER
    ::= { ipfixSelectorFunctions 2 }

psampSampCountBasedAvail OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS   read-only
    STATUS      current
    DESCRIPTION
        "This object indicates the availability of systematic
        count-based sampling at the managed node.

        A Selector may be unavailable if it is implemented but
        currently disabled due to e.g., administrative reasons, lack
        of resources or similar."
    ::= { psampSampCountBased 1 }

-- Parameter Set Table ++++++

psampSampCountBasedParamSetTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF
                PsampSampCountBasedParamSetEntry
    MAX-ACCESS   not-accessible
    STATUS      current
    DESCRIPTION
        "This table lists configurations of systematic count-based
        packet sampling. A parameter set describing a
        configuration contains two parameters: the sampling
        interval length and space."
    ::= { psampSampCountBased 2 }

psampSampCountBasedParamSetEntry OBJECT-TYPE
    SYNTAX      PsampSampCountBasedParamSetEntry
    MAX-ACCESS   not-accessible
    STATUS      current
    DESCRIPTION
        "Defines an entry in the psampSampCountBasedParamSetTable."
    INDEX { psampSampCountBasedIndex }
    ::= { psampSampCountBasedParamSetTable 1 }

PsampSampCountBasedParamSetEntry ::=
    SEQUENCE {
        psampSampCountBasedIndex      Integer32,
        psampSampCountBasedInterval    Unsigned32,
        psampSampCountBasedSpace       Unsigned32
    }

```



```
psampSampCountBasedIndex OBJECT-TYPE
    SYNTAX      Integer32 (1..2147483647)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The index of this parameter set in the
        psampSampCountBasedParamSetTable. It is used in the
        object ipfixSelectionProcessSelectorFunction entries of
        the ipfixSelectionProcessTable in the IPFIX-MIB as reference
        to this parameter set."
    ::= { psampSampCountBasedParamSetEntry 1 }

psampSampCountBasedInterval OBJECT-TYPE
    SYNTAX      Unsigned32
    UNITS       "packets"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This object specifies the number of packets that are
        consecutively sampled. A value of 100 means that 100
        consecutive packets are sampled."
    REFERENCE
        "RFC5475, Section 5.1 and RFC5477, Section 8.2"
    ::= { psampSampCountBasedParamSetEntry 2 }

psampSampCountBasedSpace OBJECT-TYPE
    SYNTAX      Unsigned32
    UNITS       "packets"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This object specifies the number of packets between two
        psampSampCountBasedInterval's. A value of 100 means that
        the next interval starts 100 packets (which are not sampled)
        after the current psampSampCountBasedInterval is over."
    REFERENCE
        "RFC5475, Section 5.1 and RFC5477, Section 8.2"
    ::= { psampSampCountBasedParamSetEntry 3 }

=====
--* Method 2: Systematic time-based Sampling
=====

-- Reference: RFC5475, Section 5.1, RFC5476 Section 6.5.2.2 and
--             RFC5477, Section 8.2
psampSampTimeBased OBJECT IDENTIFIER
    ::= { ipfixSelectorFunctions 3 }
```

```

psampSampTimeBasedAvail OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "This object indicates the availability of systematic
        time-based sampling at the managed node.

        A Selector may be unavailable if it is implemented but
        currently disabled due to e.g., administrative reasons, lack
        of resources or similar."
    ::= { psampSampTimeBased 1 }

-- Parameter Set Table ++++++

psampSampTimeBasedParamSetTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF
                PsampSampTimeBasedParamSetEntry
    MAX-ACCESS   not-accessible
    STATUS       current
    DESCRIPTION
        "This table lists configurations of systematic time-based
        packet sampling. A parameter set describing a configuration
        contains two parameters: the sampling interval length and
        the space."
    ::= { psampSampTimeBased 2 }

psampSampTimeBasedParamSetEntry OBJECT-TYPE
    SYNTAX      PsampSampTimeBasedParamSetEntry
    MAX-ACCESS   not-accessible
    STATUS       current
    DESCRIPTION
        "Defines an entry in the psampSampTimeBasedParamSetTable."
    INDEX { psampSampTimeBasedIndex }
    ::= { psampSampTimeBasedParamSetTable 1 }

PsampSampTimeBasedParamSetEntry ::=
    SEQUENCE {
        psampSampTimeBasedIndex      Integer32,
        psampSampTimeBasedInterval    Unsigned32,
        psampSampTimeBasedSpace       Unsigned32
    }

psampSampTimeBasedIndex OBJECT-TYPE
    SYNTAX      Integer32 (1..2147483647)
    MAX-ACCESS   not-accessible
    STATUS       current
    DESCRIPTION

```

```

    "The index of this parameter set in the
    psampSampTimeBasedParamSetTable. It is used in the
    object ipfixSelectionProcessSelectorFunction entries of
    the ipfixSelectionProcessTable in the IPFIX-MIB as reference
    to this parameter set."
 ::= { psampSampTimeBasedParamSetEntry 1 }

psampSampTimeBasedInterval OBJECT-TYPE
    SYNTAX      Unsigned32
    UNITS       "microseconds"
    MAX-ACCESS   read-only
    STATUS      current
    DESCRIPTION
        "This object specifies the time interval in microseconds
        during which all arriving packets are sampled."
    REFERENCE
        "RFC5475, Section 5.1 and RFC5477, Section 8.2"
    ::= { psampSampTimeBasedParamSetEntry 2 }

psampSampTimeBasedSpace OBJECT-TYPE
    SYNTAX      Unsigned32
    UNITS       "microseconds"
    MAX-ACCESS   read-only
    STATUS      current
    DESCRIPTION
        "This object specifies the time interval in microseconds
        between two psampSampTimeBasedInterval's. A value of 100
        means that the next interval starts 100 microseconds (during
        which no packets are sampled) after the current
        psampSampTimeBasedInterval is over."
    REFERENCE
        "RFC5475, Section 5.1 and RFC5477, Section 8.2"
    ::= { psampSampTimeBasedParamSetEntry 3 }

-----
--* Method 3: Random n-out-of-N Sampling
-----

-- Reference: RFC5475, Section 5.2.1, RFC5476 Section 6.5.2.3 and
--             RFC5477, Section 8.2
psampSampRandOutOfN OBJECT IDENTIFIER
    ::= { ipfixSelectorFunctions 4 }

psampSampRandOutOfNAvail OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS   read-only
    STATUS      current
    DESCRIPTION
```

"This object indicates the availability of random n-out-of-N sampling at the managed node.

A Selector may be unavailable if it is implemented but currently disabled due to e.g., administrative reasons, lack of resources or similar."

```
::= { psampSampRandOutOfN 1 }
```

-- Parameter Set Table ++++++

psampSampRandOutOfNParamSetTable OBJECT-TYPE

SYNTAX SEQUENCE OF
PsampSampRandOutOfNParamSetEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"This table lists configurations of random n-out-of-N sampling. A parameter set describing a configuration contains two parameters, the sampling size and the parent population."

```
::= { psampSampRandOutOfN 2 }
```

psampSampRandOutOfNParamSetEntry OBJECT-TYPE

SYNTAX PsampSampRandOutOfNParamSetEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"Defines an entry in the psampSampRandOutOfNParamSetTable."

INDEX { psampSampRandOutOfNIndex }

```
::= { psampSampRandOutOfNParamSetTable 1 }
```

PsampSampRandOutOfNParamSetEntry ::=

```
SEQUENCE {
    psampSampRandOutOfNIndex      Integer32,
    psampSampRandOutOfNSize      Unsigned32,
    psampSampRandOutOfNPopulation Unsigned32
}
```

psampSampRandOutOfNIndex OBJECT-TYPE

SYNTAX Integer32 (1..2147483647)

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"The index of this parameter set in the psampSampRandOutOfNParamSetTable. It is used in the object ipfixSelectionProcessSelectorFunction entries of the ipfixSelectionProcessTable in the IPFIX-MIB as reference to this parameter set."

```
::= { psampSampRandOutOfNParamSetEntry 1 }

psampSampRandOutOfNSize OBJECT-TYPE
    SYNTAX      Unsigned32
    UNITS        "packets"
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "This object specifies the number of elements taken from the
        parent Population specified in
        psampSampRandOutOfNPopulation."
    REFERENCE
        "RFC5475, Section 5.2.1 and RFC5477, Section 8.2"
    ::= { psampSampRandOutOfNParamSetEntry 2 }

psampSampRandOutOfNPopulation OBJECT-TYPE
    SYNTAX      Unsigned32
    UNITS        "packets"
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "This object specifies the number of elements in the parent
        Population."
    REFERENCE
        "RFC5475, Section 5.2.1 and RFC5477, Section 8.2"
    ::= { psampSampRandOutOfNParamSetEntry 3 }

-----
--* Method 4: Uniform probabilistic Sampling
-----

-- Reference: RFC5475, Section 5.2.2, RFC5476 Section 6.5.2.4 and
--             RFC5477, Section 8.2
psampSampUniProb OBJECT IDENTIFIER ::= { ipfixSelectorFunctions 5 }

psampSampUniProbAvail OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "This object indicates the availability of random uniform
        probabilistic sampling at the managed node.

        A Selector may be unavailable if it is implemented but
        currently disabled due to e.g., administrative reasons, lack
        of resources or similar."
    ::= { psampSampUniProb 1 }
```

```
-- Parameter Set Table ++++++

-- Reference: RFC5475, Section 5.2.2.1 and RFC5477, Section 8.2
psampSampUniProbParamSetTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF
                  PsampSampUniProbParamSetEntry
    MAX-ACCESS   not-accessible
    STATUS       current
    DESCRIPTION
        "This table lists configurations of random probabilistic
        sampling. A parameter set describing a configuration
        contains a single parameter only: the sampling probability."
    ::= { psampSampUniProb 2 }

psampSampUniProbParamSetEntry OBJECT-TYPE
    SYNTAX      PsampSampUniProbParamSetEntry
    MAX-ACCESS   not-accessible
    STATUS       current
    DESCRIPTION
        "Defines an entry in the psampSampUniProbParamSetTable."
    INDEX { psampSampUniProbIndex }
    ::= { psampSampUniProbParamSetTable 1 }

PsampSampUniProbParamSetEntry ::=
    SEQUENCE {
        psampSampUniProbIndex      Integer32,
        psampSampUniProbProbability Float64TC
    }

psampSampUniProbIndex OBJECT-TYPE
    SYNTAX      Integer32 (1..2147483647)
    MAX-ACCESS   not-accessible
    STATUS       current
    DESCRIPTION
        "The index of this parameter set in the
        psampSampUniProbParamSetTable. It is used in the
        object ipfixSelectionProcessSelectorFunction entries of
        the ipfixSelectionProcessTable in the IPFIX-MIB as reference
        to this parameter set."
    ::= { psampSampUniProbParamSetEntry 1 }

psampSampUniProbProbability OBJECT-TYPE
    SYNTAX      Float64TC
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "This object specifies the probability that a packet is
        sampled, expressed as a value between 0 and 1. The
```

probability is equal for every packet. A value of 0 means no packet is sampled since the probability is 0. A value of 1 means all packets are sampled since the probability is 1. NaN (not a number) and infinity MUST NOT be used."

REFERENCE

"RFC5475, Section 5.2.2.1 and RFC5477, Section 8.2"

::= { psampSampUniProbParamSetEntry 2 }

=====

-- Packet selection filtering methods group of objects

=====

--* Method 5: Property Match filtering

-- Reserves Method 5 (see RFC5475, Section 6.1, RFC5476

-- Section 6.5.2.5 and RFC5477)

psampFiltPropMatch OBJECT IDENTIFIER

::= { ipfixSelectorFunctions 6 }

psampFiltPropMatchAvail OBJECT-TYPE

SYNTAX TruthValue

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This object indicates the availability of property match filtering at the managed node.

A Selector may be unavailable if it is implemented but currently disabled due to e.g., administrative reasons, lack of resources or similar."

::= { psampFiltPropMatch 1 }

=====

--* Method 6: Hash filtering

=====

-- Reference: RFC5475, Section 6.2, RFC5476 Section 6.5.2.6 and

-- RFC5477, Section 8.3

psampFiltHash OBJECT IDENTIFIER ::= { ipfixSelectorFunctions 7 }

psampFiltHashAvail OBJECT-TYPE

SYNTAX TruthValue

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This object indicates the availability of hash filtering at the managed node.

A Selector may be unavailable if it is implemented but currently disabled due to e.g., administrative reasons, lack of resources or similar."

```
::= { psampFiltHash 1 }
```

```
psampFiltHashCapabilities OBJECT IDENTIFIER
```

```
::= { psampFiltHash 2 }
```

```
-- Parameter Set Table ++++++
```

```
-- Reference: RFC5475, Sections 6.2, 3.8, and 7.1
```

```
psampFiltHashParamSetTable OBJECT-TYPE
```

```
SYNTAX          SEQUENCE OF
                  PsampFiltHashParamSetEntry
```

```
MAX-ACCESS      not-accessible
```

```
STATUS          current
```

```
DESCRIPTION
```

"This table lists configurations of hash filtering. A parameter set describing a configuration contains eight parameters describing the hash function."

```
::= { psampFiltHash 3 }
```

```
psampFiltHashParamSetEntry OBJECT-TYPE
```

```
SYNTAX          PsampFiltHashParamSetEntry
```

```
MAX-ACCESS      not-accessible
```

```
STATUS          current
```

```
DESCRIPTION
```

"Defines an entry in the psampFiltHashParamSetTable."

```
INDEX { psampFiltHashIndex }
```

```
::= { psampFiltHashParamSetTable 1 }
```

```
PsampFiltHashParamSetEntry ::=
```

```
SEQUENCE {
    psampFiltHashIndex          Integer32,
    psampFiltHashFunction       INTEGER,
    psampFiltHashInitializerValue Unsigned64TC,
    psampFiltHashIpPayloadOffset Unsigned64TC,
    psampFiltHashIpPayloadSize  Unsigned64TC,
    psampFiltHashSelectedRangeMin Unsigned64TC,
    psampFiltHashSelectedRangeMax Unsigned64TC,
    psampFiltHashOutputRangeMin Unsigned64TC,
    psampFiltHashOutputRangeMax Unsigned64TC
}
```

```
psampFiltHashIndex OBJECT-TYPE
```


SYNTAX Integer32 (1..2147483647)
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
 "The index of this parameter set in the
 psampFiltHashParamSetTable. It is used in the
 object ipfixSelectionProcessSelectorFunction entries of
 the ipfixSelectionProcessTable in the IPFIX-MIB as reference
 to this parameter set."
 ::= { psampFiltHashParamSetEntry 1 }

psampFiltHashFunction OBJECT-TYPE

SYNTAX INTEGER {
 crc32(1),
 ipsx(2),
 bob(3)
}
MAX-ACCESS read-only
STATUS current
DESCRIPTION
 "The Hash Function used by this filter. The PSAMP-MIB
 defines the following Hash Functions:

 crc32(1): The CRC32 Hash Function as defined in RFC1141.

 ipsx(2): The IPSX Hash Function as described in RFC5475
 appendix A.1.

 bob(3): The BOB Hash Function as described in RFC5475
 appendix A.2.
 "
REFERENCE
 "RFC5475, Section 6.2 and Appendixes A.1 and A.2.
 RFC1141."
 ::= { psampFiltHashParamSetEntry 2 }

psampFiltHashInitializerValue OBJECT-TYPE

SYNTAX Unsigned64TC
MAX-ACCESS read-only
STATUS current
DESCRIPTION
 "This object specifies the initializer value to the hash
 function."
REFERENCE
 "RFC5475, Sections 6.2, 3.8, and 7.1"
 ::= { psampFiltHashParamSetEntry 3 }

psampFiltHashIpPayloadOffset OBJECT-TYPE

SYNTAX Unsigned64TC
MAX-ACCESS read-only
STATUS current
DESCRIPTION
 "This object specifies the IP payload offset used by a
 Hash-based Selection Selector."
REFERENCE
 "RFC5475, Sections 6.2, 3.8, and 7.1"
::= { psampFiltHashParamSetEntry 4 }

psampFiltHashIpPayloadSize OBJECT-TYPE

SYNTAX Unsigned64TC
MAX-ACCESS read-only
STATUS current
DESCRIPTION
 "This object specifies the IP payload size used by a
 Hash-based Selection Selector."
REFERENCE
 "RFC5475, Sections 6.2, 3.8, and 7.1"
::= { psampFiltHashParamSetEntry 5 }

psampFiltHashSelectedRangeMin OBJECT-TYPE

SYNTAX Unsigned64TC
MAX-ACCESS read-only
STATUS current
DESCRIPTION
 "This object specifies the value for the beginning of a hash
 function's selected range."
REFERENCE
 "RFC5475, Sections 6.2, 3.8, and 7.1"
::= { psampFiltHashParamSetEntry 6 }

psampFiltHashSelectedRangeMax OBJECT-TYPE

SYNTAX Unsigned64TC
MAX-ACCESS read-only
STATUS current
DESCRIPTION
 "This object specifies the value for the end of a hash
 function's selected range."
REFERENCE
 "RFC5475, Sections 6.2, 3.8, and 7.1"
::= { psampFiltHashParamSetEntry 7 }

psampFiltHashOutputRangeMin OBJECT-TYPE

SYNTAX Unsigned64TC
MAX-ACCESS read-only
STATUS current
DESCRIPTION

```
        "This object specifies the value for the beginning of a hash
        function's potential output range."
REFERENCE
    "RFC5475, Sections 6.2, 3.8, and 7.1"
 ::= { psampFiltHashParamSetEntry 8 }

psampFiltHashOutputRangeMax OBJECT-TYPE
    SYNTAX      Unsigned64TC
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This object specifies the value for the end of a hash
        function's potential output range."
REFERENCE
    "RFC5475, Sections 6.2, 3.8, and 7.1"
 ::= { psampFiltHashParamSetEntry 9 }

-----
-- Conformance information
-----

psampCompliances OBJECT IDENTIFIER ::= { psampConformance 1 }
psampGroups      OBJECT IDENTIFIER ::= { psampConformance 2 }

-----
-- Compliance statements
-----

psampCompliance MODULE-COMPLIANCE
    STATUS      current
    DESCRIPTION
        "The implementation of all objects is optional and depends
        on the implementation of the corresponding functionality in
        the equipment."
    MODULE -- this module
        GROUP psampGroupSampCountBased
        DESCRIPTION
            "These objects must be implemented if systematic
            count-based sampling is implemented in the equipment."
        GROUP psampGroupSampTimeBased
        DESCRIPTION
            "These objects must be implemented if systematic
            time-based sampling is implemented in the equipment."
        GROUP psampGroupSampRandOutOfN
        DESCRIPTION
            "These objects must be implemented if random n-out-of-N
            sampling is implemented in the equipment."
        GROUP psampGroupSampUniProb
```

```
DESCRIPTION
    "These objects must be implemented if uniform
    probabilistic sampling is implemented in the equipment."
GROUP psampGroupFiltPropMatch
DESCRIPTION
    "These objects must be implemented if the property match
    filtering is implemented in the equipment."
GROUP psampGroupFiltHash
DESCRIPTION
    "These objects must be implemented if hash filtering
    is implemented in the equipment."
 ::= { psampCompliances 1 }

=====
-- MIB groupings
=====

psampGroupSampCountBased OBJECT-GROUP
    OBJECTS {
        psampSampCountBasedAvail,
        psampSampCountBasedInterval,
        psampSampCountBasedSpace
    }
    STATUS current
    DESCRIPTION
        "These objects are needed if count based sampling is
        implemented."
    ::= { psampGroups 1 }

psampGroupSampTimeBased OBJECT-GROUP
    OBJECTS {
        psampSampTimeBasedAvail,
        psampSampTimeBasedInterval,
        psampSampTimeBasedSpace
    }
    STATUS current
    DESCRIPTION
        "These objects are needed if time based sampling is
        implemented."
    ::= { psampGroups 2 }

psampGroupSampRandOutOfN OBJECT-GROUP
    OBJECTS {
        psampSampRandOutOfNAvail,
        psampSampRandOutOfNSize,
        psampSampRandOutOfNPopulation
    }
    STATUS current
```

```
DESCRIPTION
    "These objects are needed if random n-out-of-N sampling is
    implemented."
 ::= { psampGroups 3 }

psampGroupSampUniProb OBJECT-GROUP
    OBJECTS {
        psampSampUniProbAvail,
        psampSampUniProbProbability
    }
    STATUS current
    DESCRIPTION
        "These objects are needed if uniform probabilistic sampling
        is implemented."
 ::= { psampGroups 4 }

psampGroupFiltPropMatch OBJECT-GROUP
    OBJECTS {
        psampFiltPropMatchAvail
    }
    STATUS current
    DESCRIPTION
        "These objects are needed if property match filtering is
        implemented."
 ::= { psampGroups 5 }

psampGroupFiltHash OBJECT-GROUP
    OBJECTS {
        psampFiltHashAvail,
        psampFiltHashFunction,
        psampFiltHashInitializerValue,
        psampFiltHashIpPayloadOffset,
        psampFiltHashIpPayloadSize,
        psampFiltHashSelectedRangeMin,
        psampFiltHashSelectedRangeMax,
        psampFiltHashOutputRangeMin,
        psampFiltHashOutputRangeMax
    }
    STATUS current
    DESCRIPTION
        "These objects are needed if hash filtering is implemented."
 ::= { psampGroups 6 }

END
```

7. Security Considerations

There are no management objects defined in this MIB module that have a MAX-ACCESS clause of read-write and/or read-create. So, if this MIB module is implemented correctly, then there is no risk that an intruder can alter or create any management objects of this MIB module via direct SNMP SET operations.

All tables in this MIB module may be considered sensitive or vulnerable in some network environments because objects in the tables may reveal information about the network infrastructure and device configuration. It is thus important to control even GET and/or NOTIFY access to these objects and possibly to even encrypt the values of these objects when sending them over the network via SNMP.

SNMP versions prior to SNMPv3 did not include adequate security. Even if the network itself is secure (for example by using IPsec), there is no control as to who on the secure network is allowed to access and GET/SET (read/change/create/delete) the objects in this MIB module.

It is RECOMMENDED that implementers consider the security features provided by the SNMPv3 framework (see [RFC3410], section 8), including full support for the SNMPv3 cryptographic mechanisms (for authentication and privacy).

Further, deployment of SNMP versions prior to SNMPv3 is NOT RECOMMENDED. Instead, it is RECOMMENDED to deploy SNMPv3 and to enable cryptographic security. It is then a customer/operator responsibility to ensure that the SNMP entity giving access to an instance of this MIB module is properly configured to give access to the objects only to those principals (users) who have legitimate rights to GET or SET (change/create/delete) them.

8. IANA Considerations

The MIB module in this document uses the following IANA-assigned OBJECT IDENTIFIER values recorded in the SMI Numbers registry:

Descriptor	OBJECT IDENTIFIER value
-----	-----
psampMIB	{ mib-2 xxx }

Further on, IANA will register the following toplevel OIDs in the IPFIX-SELECTOR-MIB Functions sub-registry at <http://www.iana.org/assignments/smi-numbers> according to the procedures set forth in [RFC6615]:

Decimal	Name	Description	Reference
2	psampSampCountBased	Systematic Count-based Sampling	[RFCyyyy]
3	psampSampTimeBased	Systematic Time-based Sampling	[RFCyyyy]
4	psampSampRandOutOfN	Random n-out-of-N Sampling	[RFCyyyy]
5	psampSampUniProb	Universal Probabilistic Sampling	[RFCyyyy]
6	psampFiltPropMatch	Property Match Filtering	[RFCyyyy]
7	psampFiltHash	Hash-based Filtering	[RFCyyyy]

The prerequisites set forth for addition of these OIDs are to be verified based on the content of this document.

Editor's Note (to be removed prior to publication): the IANA is requested to assign a value for "xxx" under the 'mib-2' subtree and to record the assignment in the SMI Numbers registry. When the assignment has been made, the RFC Editor is asked to replace "xxx" (here and in the MIB module) with the assigned value and to remove this note. The RFC editor is also asked to replace "yyyy" in this document and the MIB module by the number of the RFC when the assignment has been made.

9. Acknowledgment

This document is a product of the PSAMP and IPFIX working groups. The authors would like to thank the following persons: Paul Aitken for his detailed review, Dan Romascanu and the MIB doctors, and many more, for the technical reviews and feedback.

10. References

10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2578] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.
- [RFC2579] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Textual Conventions for SMIv2", STD 58, RFC 2579, April 1999.
- [RFC2580] McCloghrie, K., Perkins, D., and J. Schoenwaelder, "Conformance Statements for SMIv2", STD 58, RFC 2580, April 1999.

- [RFC2564] Kalbfleisch, C., Krupczak, C., Presuhn, R., and J. Saperia, "Application Management MIB", RFC 2564, May 1999.
- [RFC5101] Claise, B., "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information", RFC 5101, January 2008.
- [RFC5477] Dietz, T., Claise, B., Aitken, P., Dressler, F., and G. Carle, "Information Model for Packet Sampling Exports", RFC 5477, March 2009.
- [RFC6615] Dietz, T., Kobayashi, A., Claise, B., and G. Muenz, "Definitions of Managed Objects for IP Flow Information Export", RFC 6615, June 2012.
- [RFC6340] Presuhn, R., "Textual Conventions for the Representation of Floating-Point Numbers", RFC 6340, August 2011.

10.2. Informative References

- [RFC3410] Case, J., Mundy, R., Partain, D., and B. Stewart, "Introduction and Applicability Statements for Internet-Standard Management Framework", RFC 3410, December 2002.
- [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.
- [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.

Authors' Addresses

Thomas Dietz (editor)
NEC Europe Ltd.
NEC Laboratories Europe
Kurfuersten-Anlage 36
Heidelberg 69115
DE

Phone: +49 6221 4342-128
Email: dietz@neclab.eu

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

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

Juergen Quittek
NEC Europe Ltd.
NEC Laboratories Europe
Kurfuersten-Anlage 36
Heidelberg 69115
DE

Phone: +49 6221 4342-115
Email: quittek@neclab.eu

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 10, 2013

H. Scholz
VOIPFUTURE GmbH
July 9, 2012

RTP Stream Quality Information Export using IPFIX
draft-scholz-ipfix-rtp-audio-quality-00

Abstract

This draft defines a set of Information Elements and matching Templates to convey RTP media stream quality information in IPFIX packets. The Information Elements describe the RTP quality using the R-factor and Mean Opinion score (MOS) for the entire duration of a monitored stream or for a smaller time slice thereof.

Status of this Memo

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

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

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

This Internet-Draft will expire on January 10, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1.	Introduction	3
1.1.	Use Cases	3
1.1.1.	Quality of Service (QoS) Monitoring	3
1.1.2.	Service Level Agreement (SLA)	3
1.1.3.	Troubleshooting	3
2.	Conventions	4
3.	MOS measurement	4
3.1.	rtpMOSAlg	4
3.2.	rtpMOSClass1	5
3.3.	rtpMOSClass2	5
3.4.	rtpMOSClass3	5
3.5.	rtpMOSClass4	6
3.6.	rtpMOSClass5	6
3.7.	rtpMinMOS	6
3.8.	rtpAvgMOS	7
3.9.	rtpMaxMOS	7
3.10.	rtpMinRFactor	7
3.11.	rtpAvgRFactor	7
3.12.	rtpMaxRFactor	8
4.	Recommended Templates	8
4.1.	Entire stream	8
4.2.	Time slice	8
5.	Examples	8
6.	Acknowledgements	9
7.	IANA Considerations	9
8.	Security Considerations	9
9.	References	9
9.1.	Normative References	9
9.2.	Informative References	9
	Author's Address	10

1. Introduction

IPFIX [RFC5101] and [RFC5102] define a framework allowing to export arbitrary data from so called IPFIX exporters. One type of IPFIX exporter may be co-located with Session Initiation Protocol (SIP) [RFC3261] based VoIP entities or passively observe SIP based VoIP calls. The signaling messages can be exported using [I-D.trammell-ipfix-sip-msg] and Real Time Protocol (RTP) [RFC3550] media streams are covered in [I-D.akhter-ipfix-perfmon]. Media quality is out of the scope of both these documents. This document defines a set of additional IPFIX Information Elements (IEs) to describe RTP audio stream quality.

1.1. Use Cases

RTP stream flow information contained in IPFIX flow records can be used for various tasks such as Quality of Service (QoS) monitoring, Service Level Agreement (SLA) validation and general troubleshooting of VoIP networks.

1.1.1. Quality of Service (QoS) Monitoring

Aggregated to higher-level metrics the in-depth information provided by the RTP (and optionally SIP) flow records allow service providers to gauge the overall quality of their network in terms of the quality of experience (QoE). On this level an individual call is less important but the overall quality (e.g. amount of minutes meeting certain quality standards) can be used to get a quick overview on the network and service performance.

1.1.2. Service Level Agreement (SLA)

SLAs are typically used as part of contracts between two network operators. The requirements on the reliability of the data may be higher compared to QoS Monitoring as the failure to meet contractually agreed quality standards often has a direct commercial impact.

1.1.3. Troubleshooting

An active network component (SIP proxy, B2BUA, media server) may not have the capabilities to store session related information for a long time to facilitate troubleshooting capabilities (e.g. due to missing hard-disk). Such a system or a group of systems may run the metering process and export the data to a collector for processing or troubleshooting purposes.

2. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

3. MOS measurement

A multitude of Mean Opinion Score (MOS) assessment algorithms have been defined of which only one or few may be available to an IPFIX Metering Process. The quality (i.e. accuracy) of these algorithms varies and has to be noted when transporting MOS values.

An IPFIX Metering Process may use these Information Elements to convey information on the duration of the stream in which the quality fell into the respective category as well as the measurement algorithm used to obtain the information.

3.1. rtpMOSCAlg

The values carried in this IE are taken from the "RTCP XR QoE metric block - Calculation Algorithm" sub-registry of the "RTP Control Protocol Extended Reports (RTCP XR) Block Type Registry" as defined in [I-D.wu-xrblock-rtcp-xr-quality-monitoring].

Even when an algorithm other than G.107 is used the rtpMOSClassN Information Elements use the R-Factor based classes as defined in the G.107 documentation.

Description: The calculation algorithm (CAlg) used by the Metering Process to calculate the MOS value.

0: undefined: The algorithm is not known/specified.

1: ITU-T P.564

2: G.107

3: G.107 / ETSI TS 101 329-5 Annex E

4: ITU-T P.NAMS

5: ITU-T P.NBAMS

6: RTCP - Real Time Control Protocol (not defined in registry!)

Data Type: unsigned8

Data Type Semantics: identifier

PEN (provisional): tbd

ElementId (provisional): tbd

The MOS values calculated are separated into MOS classes based on the ITU-T G.107 classes.

3.2. rtpMOSClass1

Description: Number of seconds the monitored stream had a MOS quality lower than 3.10

Data Type: float32

Data Type Semantics: deltaCounter

PEN (provisional): tbd

ElementId (provisional): tbd

3.3. rtpMOSClass2

Description: Number of seconds the monitored stream had a MOS quality larger than or equal 3.10 and lower than 3.60

Data Type: float32

Data Type Semantics: deltaCounter

PEN (provisional): tbd

ElementId (provisional): tbd

3.4. rtpMOSClass3

Description: Number of seconds the monitored stream had a MOS quality larger than or equal 3.60 and lower than 4.03

Data Type: float32

Data Type Semantics: deltaCounter

PEN (provisional): tbd

ElementId (provisional): tbd

3.5. rtpMOSClass4

Description: Number of seconds the monitored stream had a MOS quality larger than or equal 4.03 and lower than 4.34

Data Type: float32

Data Type Semantics: deltaCounter

PEN (provisional): tbd

ElementId (provisional): tbd

3.6. rtpMOSClass5

Description: Number of seconds the monitored stream had a MOS quality larger than or equal 4.34

Data Type: float32

Data Type Semantics: deltaCounter

PEN (provisional): tbd

ElementId (provisional): tbd

3.7. rtpMinMOS

Description: Minimum MOS value measured in the monitoring interval.

Data Type: float32

Data Type Semantics: quantity

PEN (provisional): tbd

ElementId (provisional): tbd

3.8. rtpAvgMOS

Description: Average MOS value measured in the monitoring interval.

Data Type: float32

Data Type Semantics: quantity

PEN (provisional): tbd

ElementId (provisional): tbd

3.9. rtpMaxMOS

Description: Maximum MOS value measured in the monitoring interval.

Data Type: float32

Data Type Semantics: quantity

PEN (provisional): tbd

ElementId (provisional): tbd

3.10. rtpMinRFactor

Description: Minimum R-Factor measured in the monitoring interval.

Data Type: float32

Data Type Semantics: quantity

PEN (provisional): tbd

ElementId (provisional): tbd

3.11. rtpAvgRFactor

Description: Average R-Factor measured in the monitoring interval.

Data Type: float32

Data Type Semantics: quantity

PEN (provisional): tbd

ElementId (provisional): tbd

3.12. rtpMaxRFactor

Description: Maximum R-Factor measured in the monitoring interval.

Data Type: float32

Data Type Semantics: quantity

PEN (provisional): tbd

ElementId (provisional): tbd

4. Recommended Templates

The defined RTP stream IPFIX templates must support both IPv4 and IPv6 transport. They need to carry either flow information regarding the entire duration of an RTP stream or specific to a shorter observation interval.

The template incorporates IEs from [I-D.akhter-ipfix-perfmon] to describe the RTP stream.

In order to correlate the RTP quality information with signaling information (e.g. subscriber IDs) a correlation ID may be added to the template. Note that this ID has yet to be defined and is outside the scope of this document.

4.1. Entire stream

tbd

4.2. Time slice

tbd, based on previous template. Split a single RTP stream in three flow records as example including (empty) 'RTP stream ended' flow record.

5. Examples

tbd

6. Acknowledgements

tbd

7. IANA Considerations

tbd

8. Security Considerations

tbd

9. References

9.1. Normative References

- [I-D.wu-xrblock-rtcp-xr-quality-monitoring]
Hunt, G., Clark, A., Wu, W., Schott, R., and G. Zorn,
"RTCP XR Blocks for QoE metric reporting",
draft-wu-xrblock-rtcp-xr-quality-monitoring-06 (work in
progress), December 2011.
- [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.

9.2. Informative References

- [I-D.akhter-ipfix-perfmon]
Akhter, A., "Information Elements for Flow Performance
Measurement", draft-akhter-ipfix-perfmon-00 (work in
progress), October 2010.
- [I-D.trammell-ipfix-sip-msg]
Claise, B., Trammell, B., Kaplan, H., and S. Niccolini,
"SIP Message Information Export using IPFIX",
draft-trammell-ipfix-sip-msg-02 (work in progress),
October 2011.

- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

Author's Address

Hendrik Scholz
VOIPFUTURE GmbH
Wendenstrasse 4
Hamburg 20097
Germany

Phone: +49 40 688 900 100
Email: hscholz@voipfuture.com
URI: <http://www.voipfuture.com/>

