IPFIX Working Group A. Akhter
Internet-Draft Cisco Systems
Intended status: Standards Track October 25, 2010
Expires: April 28, 2011

Information Elements for Flow Performance Measurement
draft-akhter-ipfix-perfmon-01.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

Copyright Notice

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 Agreemnt (SLA) Validation 6
3.3. Fault Isolation and Troubleshooting 6
4. New Information Elements 6
4.1. Transport Layer 7
4.1.1. perfPacketLoss 7
4.1.2. perfPacketExpected 9
4.1.3. perfPacketLossRate 10
4.1.4. perfPacketLossEvent 11
4.1.5. perfPacketInterArrivalJitterAvg 12
4.1.6. perfPacketInterArrivalJitterMin 14
4.1.7. perfPacketInterArrivalJitterMax 15
4.2. User and Application Layer 16
4.2.1. perfSessionSetupDelay 16
4.3. Contextual Elements 17
4.3.1. mediaRTPSSRC 17
4.3.2. mediaRTPPayloadType 18
4.3.3. mediaCodec 19
5. Security Considerations 20
6. IANA Considerations 20
7. References 20
7.1. Normative References 20
7.2. Informative References 21
Author’s Address 22

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 sensitive 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 in an attempt to wring 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.

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.

Where possible, an attempt has been made to make use of existing definitions of metrics ([RFC4710]) and if needed, clarify and expand on them to widen their usage with additional applications. The methodology described in [I-D.ietf-pmol-sip-perf-metrics] is used to describe the methodology of measurement. As this document also 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 informationElementDataTypeas 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.

Use and Applications An explanation of how this particular information element would be used.

Calculation Method: In the case of metrics, this section describes how the metric is calculated, as well as any special conditions.

Units of Measurement: In the case of metrics, what are the units of measurement. The text here is expected to be wider and more descriptive than in the IPFIX Element Units section.

Measurement Timing: Discussion on the acceptable range of timing and sampling intervals.

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).

3.2. Service Level Agreemnt (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 that 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 date. 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: 0xFFFFFFFE

Element Id: TBDperfPacketLoss

Status: current

Use and Applications The packet loss metric can be used to determine if there is a network impairment that is causing packet loss upstream of the measurement point. When there are observation points on either side of the impairment location it is possible to locate the impairment. With the location information the operator can is able to perform quicker fault-isolation as well as shorten time to resolution.

Calculation Method: This metric requires that each IP packet be individually marked with a monotonically incrementing sequence number. A number of encapsulations support this type of sequencing: IPSec ESP [RFC4303], GRE [RFC2890] and RTP [RFC3550]. An analysis of the sequence number field can yield the lost number of packets. In certain cases, there might be an element of discovery and synchronization of the flow itself before the measurement can be made. An example of this can be found for RTP flows running on ephemeral UDP port numbers. In these cases, reporting 0 as packet loss would be misleading and the value 0xFFFFFFFF MUST be used in cases where the packet loss value cannot be determined. In the case of a monitor interval where synchronization was achieved mid-interval, the loss packet counter MAY be used to represent the remainder of the interval. As this metric is a deltaCounter, the number of loss packets only represent the observation within the reporting interval. Due to the dependency on the arrival of a packet with a sequence number to calculate loss, the loss calculation may be indefinitely delayed if no more packets arrive at all. For the case of RTP, in addition to the 16 bit sequence number field in RFC3550, there is also the additional 16-bit high-order sequence number field (for a total of 32-bit seq number space) that is used in RFC3497 [RFC3497]. RFC3497 traffic runs at a very high rate and the 32-bit field allow for additional time for wrapping (21 seconds). So, a loss span of greater than 21 seconds measured only by the 16-bit field will lead to inaccurate reporting. In the case of secure RTP [RFC3711], the relevant portion of the RTP header is in the clear and lost packet counting can still be performed. It is important to note that the sequence number space is unique per RTP SSRC. Therefore it is important to track the high sequence number seen on a per SSRC-5-tuple basis. There may be multiple SSRCs in a single 5-tuple. Certain applications inject non-RTP traffic into the same 5-tuple as the media stream. RTCP packets may be seen in the same 5-tuple as the RTP stream [RFC5761], and STUN [RFC5389] packets may also be seen. The loss detection should ignore these packets. There may be spans within the network where header compression schemes such as [RFC2508] are used. In cases where the measurement device is terminating the compression, and the measurement implementation does not support calculation of the metric the value 0xFFFFFFFF MUST be reported. In other cases the measurement point may be at a midpoint of the header compression network span. Depending on the mechanics of header compression, sequencing information may be present and it is possible to calculate the metric. In such cases the implementation SHOULD perform the calculation and report the metric.

Units of Measurement: packets

Measurment Timing To be able to calculate this metric a continuous set of the flow’s packets (as each would have an incrementing sequence number) needs to be monitored. Therefore, per-packet sampling would prevent this metric from being calculated. However, there are other sampling methodologies that might be usable. It is possible to generate sampled metrics by sampling spans of continuous packets, however a portion of the span may have to be utilized for resynchronization of the sequence number. Another form of acceptable sampling would be at the flow level.

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: 0xFFFFFFFE

Element Id: TBDperfPacketExpected

Status: current

Use and Applications The perfPacketExpected is a mid-calculation metric used in the calculation of perfPacketLossRate.

Calculation Method: The subtraction of the last sequence number from the first sequence number in monitoring interval yields the expected count. As discussed with perfPacketLost, there might be a delay due to synchronization with the flow’s sequence numbers and in such times the value of the metric should be set to 0xFFFFFFFE. Care has to be taken to account for cases where the packet’s sequence number field wraps. For RTP, the expected count calculation formula can be found in Appendix A.3 of [RFC3550]. Refer to the perfPacketLoss metric regarding considerations for header compression. The value 0xFFFF is used to represent cases where the metric could not be calculated.

Units of Measurement: packets

Measurment Timing Same considerations as perfPacketLoss

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

Use and Applications The perfPacketLossRate metric can be used to normalize the perfPacketLoss metric to handle cases where different flows are running at different packet per second (PPS) rates. Due to the normalization, comparisons can now be made against thresholds (for creating alerts, etc.). In addition, the percentage form of the metric allows for comparisons against other flows at the same observation point to determine if there is an equal bias for drops between the flows. Otherwise, the perfoPacketLossRate is used in same way as perfPacketLoss.

Calculation Method: The number of lost packets divided by the number of expected packets in an interval period multiplied by 100. In cases where perfPacketLoss is unknown (for example due to synchronization issues), the perfPacketLossRate would also be unknown. In such cases perfPacketLossRate MUST be set to 0xFFFF. If there are multiple flows whose loss rate is being aggregated, then the average of the individual flows is used. Refer to the perfPacketLoss metric regarding considerations for header compression. The value 0xFFFF is used to represent cases where the metric could not be calculated.

Units of Measurement: percentage

Measurment Timing Same notes as perfPacketLossRate

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: packets

Element Range Begin: 0

Element Range End: 0xFFFFFFFE

Element Id: TBDperfPacketLossEvent

Status: current

Use and Applications The perfPacketLossEvent metric can provide loss information for protocols that do not implement per packet sequencing. Similarly to the perfPacketLoss metric, the packet loss event metric can be used to determine if there is a network impairment that is causing packet loss upstream of the measurement point. In cases where both the perfPacketLoss and perfPacketLossEvent metric are available, the ratio between the packet loss and packet event count can provide the average loss length. The average loss length provides additional information regarding the cause of the loss. For example, a dirty fiber connection might have a low average loss length, while a routing protocol convergence will have a high loss length.

Calculation Method: This data value is a simplified version of the Lost Packets metric. Whereas Lost Packets counts individual packet loss, the 'loss event count' metric counts sets of packets that are lost. For example, in the case of a sequence of packets: 1,3,6,7,10 the packets marked 2,4,5,8 and 9 are lost. So, a total of 5 packets are lost. This same sequence translates to 3 loss events: (2), (4,5) and (8,9). In the case of RTP, the sequence number in the RTP header can be used to identify loss events. Certain protocols such as TCP and UDP+MPEG2-TS encapsulation in IP have sequencing information, but the sequence field is incremented by individual IP packets. As a side note, in the case of UDP+ MPEG2-TS encapsulation the simple use of RTP+MPEG2-TS via [RFC2250] results in the avaliability of the more granular perfPacketLoss metrics. In these cases, the perfPacketLoss metric cannot be calculated but the perfPacketLossEvent can be calculated and can provide detection of loss. The value 0xFFFFFFFF is used to represent non-applicable cases such as lack of sequence number synchronization. Many of the same considerations as for perfPacketLoss apply to perfPacketLoss event. Please refer to the Calculation Method section of the perfPacketLoss.

Units of Measurement: event counts

Measurment Timing Please refer to the measurement timing section of perfPacketLoss.

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: 0xFFFFFFFE

Element Id: TBDperfPacketInterArrivalJitterAvg

Status: current

Use and Applications The inter arrival jitter data value can be used be network operator to determine the network's impact to the spacing in between a media stream's packets as they traverse the network. For example, in the case of media applications, the receiving end system is expecting these packets to come in at a particular periodicity and large deviations may result in de-jitter buffers adding excessive delay, or the media packets being discarded. When the data is reported from multiple intermediate nodes, the area of the network that is having a detrimental contribution can be identified. On a non-media application level, the inter arrival jitter metrics can be used for early indication queuing contention within the network (which could lead to packet loss).

Calculation Method: The inter arrival jitter value makes use of the association of sending time with an IP packets and comparison of the arrival time on the monitoring point. In certain protocols, a representation of sending time is encoded into the header itself. For example, in the case of RTP packets, the RTP header's timestamps field represents encoder clock ticks-- which are representations of time. Similarly, in the case of TCP options encode absolute timestamps values. For RTP the calculation method can be found in Appendix A of [RFC3550]. It should be noted that the RFC3550 calculation is on the last 16 packets measured. The most recent value calculated SHOULD be reported at the end of the monitoring interval. The range of the jitter values during the monitoring interval can be reported using perfPacketInterArrivalJitterMin and perfPacketInterArrivalJitterMax. Similarly to the perfPacketLoss case there may be periods of time where the jitter value cannot be calculated. In these cases, the 0xFFFFFFFF value should be used to convey the lack of availability of the metric. As mentioned earlier, the RTP header timestamps is actually a 'sample-stamp' (ie clicks) from the encoder's clock. The frequency of the clock is dependent on the codec. Some codecs (eg AAC-LD) support multiple possible frequencies one of which is then selected for the media-stream. The mapping to clock rate can be performed via mapping from the static RTP payload type (RTP-PT), but newer codecs are make use of the dynamic payload type range and the RTP-PT (in the dynamic case) cannot be used to determine the clock frequency. There are various methods by which the clock frequency (deep packet inspection of the signalling, manual configuration, etc.) can be associated to the calculation method. The frequency should be locked in the metering layer to a unique combination of the IP source, IP destination, IP protocol layer-4 ports, RTP-PT and SSRC. By strict RFC3550 definition, the SSRC is set to a specific encoder clock and it is the SSRC that should be tracked rather than payload type. However, in recent discussions it has been noted that there are RTP implementations that might change the encoder clock frequency while maintaining the SSRC value. An encoder frequency change will be accompanied by a different RTP-PT.

Units of Measurement: microseconds

Measurment Timing Please refer to the measurement timing section of perfPacketLoss.

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: 0xFFFFFFFE

Element Id: TBDperfPacketInterArrivalJitterMin

Status: current

Use and Applications Please refer to the 'Use and Applications' section of perfPacketInterArrivalJitterAvg. This specific metric, along with perfPacketInterArrivalJitterMax, is to capture the range of measurements observed within a monitoring interval as the average function may hide extremes.

Calculation Method: Please see the perfPacketInterArrivalJitterAvg section for general calculation section. The average calculation is evaluated on a running basis over the last 16 packets and the entire monitoring interval is not covered. In this metric, the minimum value is taken over the entire monitoring interval.

Units of Measurement: microseconds

Measurment Timing Please refer to the measurement timing section of perfPacketLoss.

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: 0xFFFFFFFE

Element Id: TBDperfPacketInterArrivalJitterMax

Status: current

Use and Applications Please refer to the 'Use and Applications' section of perfPacketInterArrivalJitterAvg. This specific metric, along with perfPacketInterArrivalJitterMin, is to capture the range of measurements observed within a monitoring interval as the average function may hide extremes.

Calculation Method: Please see the perfPacketInterArrivalJitterAvg section for general calculation section. The average calculation is evaluated on a running basis over the last 16 packets and the entire monitoring interval is not covered. In this metric, the maximum value is taken over the entire monitoring interval.

Units of Measurement: microseconds

Measurment Timing Please refer to the measurement timing section of perfPacketLoss.

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: 0xFFFFFFFE

Element Id: TBDperfSessionSetupDelay

Status: current

Use and Applications The session setup delay metric can measure the end user initial wait experience as seen from the network transaction level. The value will not only include the network flight time, but also includes the server response time and may be used to alert the operator in cases where the overall service is overloaded and thus sluggish, or within normal operating values.

Calculation Method: Measure distance in time between the first bit of request and the first bit of the response. For the case of SIP, please see Section 4.3.1 of [I-D.ietf-pmol-sip-perf-metrics]

Units of Measurement: microseconds

Measurment Timing This measurement can be sampled on a session by session basis. It may be advisable to set sample targets on a per source range - to destination basis. Due to the nature of measurement intervals, there may be a period of time (and thus measurement reports) in which the perfSessionSetupDelay value has not been calculated. In these cases the value 0xFFFFFFFE MUST be used and can be interpreted to mean not applicable. For measurement intervals after perfSessionSetupDelay has been calculated and the existing calculated perfSessionSetupDelay value SHOULD be sent if reporting only on that single session. However, if multiple sessions are summarized in the report then the average for perfSessionSetupDelay values calculated in the most recent interval SHOULD be used. The intention with this behavior is to acknowledge that the value has not bee calculated, and when it has provide the freshest values available.

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 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: 0xFFFFFFFE

Element Id: TBDmediaRTPSSRC

Status: current

Use and Applications The RTP SSRC value denotes a specific media stream. As such when trying to differentiate media stream problems between session participants the SSRC field is needed.

Calculation Method: Copy from RTP header’s SSRC field as defined in [RFC3550]. In the case of a non-RTP flow, or the time period in which the flow has not been verified to be a RTP flow the value 0xFFFFFFFE MUST be reported.

Units of Measurement: identifier

Measurment Timing It is possible that the SSRC may have be renegotiated mid-session due to collisions with other RTP senders.

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 on the flow path or within the endpoints.

Element Data Type: unsigned16

Element Semantics: identifier

Element Units: octets

Element Range Begin: 0

Element Range End: 0xFF

Element Id: TBDmediaRTPPayloadType

Status: current

Use and Applications The RTP PT conveys the payload format and media encoding used in the RTP payload. For simple cases, where the RTP PT is from the statically defined range this can lead to an understanding of type of media codec used. With the knowledge of the codec being used the degree of media impairment (given loss values and jitter) can be estimated better. However, for more recent codecs, the RTP dynamic range is used. In these cases the RTP payload values are dynamically negotiated. In the case of a non-RTP flow, or the time period in which the flow has not been verified to be a RTP flow, the value 0xFFFF MUST be reported.

Calculation Method: Copy from RTP header’s RTP-PT field as defined in [RFC3550]

Units of Measurement: identifier

Measurment Timing

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

Use and Applications The media codec value conveys the name of the codec used to encode the media in the flow being monitored. Simply reporting loss and jitter measurements are useful for detection of network problems. However, judging the degree of the impact on the audio/video experience needs additional information. The most basic information is the codec being used which when coupled with per-codec knowledge of sensitivity to the transport metrics a better idea of the experience can be gained.

Calculation Method: The valid values for the mediaCodec are listed on the IANA media-types registry. Analysis of the RTP payload type may lead to the determination of the media codec. However, with the use of the RTP dynamic payload type range the media information is not encoded into the data packet. For these cases, intermediate nodes may need to perform inspection of the signalling (SIP, H.323, RTSP, etc.). In cases where the mediaCodec cannot be determined, the value ’unknown’ MUST be used.

Units of Measurement: identifier

Measurment Timing

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. References

7.1. 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)".

7.2. 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-05 (work in progress), October 2010.

[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-00 (work in progress), October 2010.

[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] "probe - 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 B. Claise
Internet-Draft P. Aitken
Intended Status: Informational N.ir Ben-Dvora
Expires: April 16, 2011 Cisco Systems, Inc.
October 16, 2010

Export of Application Information in IPFIX
draft-claise-export-application-info-in-ipfix-00

Status of this Memo

Abstract

This document specifies an extension to the IP Flow Information eXport (IPFIX) protocol specification in [RFC5101] and 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].

Table of Contents

1. Overview.. 4
   1.1. IPFIX Documents Overview........................... 4
2. Introduction.. 4
2.1. Application Information Use Case...................... 6
3. Terminology... 7
   3.1. New Terminology.................................... 7
4. applicationTag Information Element Specification......... 7
   4.1. Existing Classification Engine IDs.................. 9
   4.2. Options Template Record for the Application Name... 11
   4.3. Resolving IANA L4 port collisions.................. 11
5. Application Tag Examples............................... 13
   5.1. Example 1: Layer 2 Protocol........................ 13
   5.2. Example 2: Standardized IANA Layer 3 Protocol...... 14
   5.3. Example 3: Cisco Systems Proprietary Layer 3 Protocol15
   5.4. Example 4: Standardized IANA Layer 4 Port.......... 17
   5.5. Example 4: Layer 7 Application..................... 18
   5.6. Example: port Obfuscation.......................... 19
   5.7. Example: Application Mapping Options Template...... 20
6. IANA Considerations.................................. 21
   6.1. applicationDescription............................. 21
   6.2. applicationTag..................................... 22
   6.3. applicationName.................................... 22
7. Security Considerations............................... 22
8. References.. 22
   8.1. Normative References............................... 22
   8.2. Informative References............................. 23
9. Acknowledgement...................................... 23
10. Authors' Addresses.................................. 24
Appendix A. Additions to XML Specification of IPFIX
Information Elements.................................... 25

List of Figures

Figure 1: applicationTag Information Element 8
Figure 2: Selector ID encoding 9
Table 1: Existing Classification Engine IDs 10
Table 2: IANA layer 4 port collisions 12
Table 3: Resolving layer 4 UDP ports 12

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 [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 the user processes that exchange packets between them (such as the web applications, peer to peer applications, file transfer, e-mail applications, etc.)

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

1. L2 protocols (such as ARP, PPP, LLDP)
2. IP protocols (such as ICMP, IGMP, GRE)
3. TCP or UDP ports (such as HTTP, Telnet, FTP)
4. Application headers
5. Packet content signatures
6. Traffic behavior

The exact application identification methods are part of the Metering Process internals that aims to provide an accurate identification with a minimum 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 run on different port than the IANA assigned one. For example a HTTP server might run a TCP port 23 (assigned to telnet in [IANA-PORTS])
2. IANA does 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 signatures become more and more complex

For that reason, such Metering Processes usually detect application based on multiple mechanisms in parallel. Detecting applications based only on port matching might wrongly identify the traffic. Note that this example stresses the need for the engine strength. If the Metering Process is capable of detecting applications more accurately it is considered as stronger and more accurate.

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

Cisco Systems uses the IPFIX application tag as described in section 4. to export the application information with the IPFIX protocol [RFC5101].

Application could be defined at different OSI layers, from the layer 2 to the layer 7. Examples: Cisco Discovery Protocol is layer 2 application, ICMP is layer 3 application [IANA-PROTO], HTTP is layer 4 application [IANA-PORTS], and skype is 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 as the some applications require well known specifications. Therefore, some reverse engineering is required, as well as a ubiquitous language for application signature. Clearly not realistic.

As this specification focuses on the application information encoding, this document doesn't contain an application registry for non IANA applications. However, a reference to the Cisco assigned numbers can be found at [CISCO].

2.1. Application Information Use Case

There are several use cases on which the application information is used:

1. Network Visibility

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

2. Billing Services

   In some cases, network providers are willing to bill different applications differently. For example, provide different billing for VoIP and Web browsing.

3. Congestion Control

   While the traffic demand is increasing (mainly due to the high usage of peer to peer applications, video applications and web download applications), the providers revenue doesn't grow. Providers are looking at a more efficient way to control and prioritize the network utilization. An application aware bandwidth control system is used to prioritize the traffic based on the applications, giving the critical applications priority over the non-critical applications.

4. 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 of 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]. 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 Tag

A unique identifier for an application. The Application Tag consists of a Classification Engine ID and a Selector ID [RFC5476].

4. applicationTag Information Element Specification

This document specifies the applicationTag Information Element, which is composed of two parts:

1. 8 bits of Classification Engine ID.
2. m bits of Selector ID. The Selector ID length varies depending on the engine.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Class. Eng. ID|         Selector ID  ...                      |
```

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...                                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: applicationTag 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.

Selector ID

A unique identifier of the application for a specific Classification Engine ID.

Note that the Selector ID term is in sync with the PSAMP terminology. See [RFC5476], Packet Sampling (PSAMP) Protocol Specifications.

When an application is detected, the most granular application is encoded in the Application Tag: for example, ICMP would be encoded as layer 3 value 1, SNMP as layer 4 value 161, bittorent as layer 7 value 69.

The overall length of the applicationTag Information Element may be specified either in the IPFIX Template Record or by using an IPFIX Variable-Length Information Element. The receiver / decoder must respect this length rather than using the Classification Engine ID to make an assumption about the Selector ID size.

When exporting applicationTag information in IPFIX, the applicationTag SHOULD be encoded in a variable-length Information Element [RFC5101]. However, if a legacy protocol such as NetFlow version 9 is used, and this protocol doesn’t support variable length Information Elements, then either multiple templates (one per applicationTag length), or a single template corresponding to the maximum sized applicationTag MUST be used. This avoids the need for multiple Template Records with different applicationTag lengths when the IPFIX variable length encoding [RFC5101] is not available.

As a consequence, although some Application Tags can be encoded in a smaller number of bytes (eg, an IANA L3 protocol encoding would take 2 bytes, while an IANA L4 port encoding would take 3 bytes), nothing prevents an Exporting Process from exporting all Application Tags with a larger fixed length.

Note that the Selector ID value is always encoded in the least significant bits as shown:

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Class. Eng. ID |            zero-valued upper-bits ...         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     ...  Selector ID                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: Selector ID encoding

4.1. Existing Classification Engine IDs

The following Engine IDs have been allocated by Cisco Systems.

Name	Value	Description
	0	Invalid.
IANA-L3	1	The IANA protocol (layer 3) number is exported in the Selector ID. See http://www.iana.org/assignments/protocol-numbers.
CANA-L3	2	Cisco Systems proprietary layer 3 definition. Cisco Systems can still export its own layer 3 protocol numbers, while waiting for IANA to assign it. The Selector ID has a global significance for all Cisco Systems devices under CANA governance. Hopefully the same IDs will be maintained after the IANA standardization.
IANA-L4	3	IANA layer 4 well-known port number is exported in the Selector ID. See http://www.iana.org/assignments/port-numbers. Note: as a flow is unidirectional, it contains the destination port in a flow from the client to the server.
CANA-	4	Cisco Systems proprietary layer 4

L4		definition. Cisco Systems can still export its own layer 4 port numbers, while waiting for IANA to assign it. The Selector ID has global significance for all Cisco Systems devices under CANA governance. Hopefully the same ID will be maintained after the IANA standardization. Example: IPFIX had the port 4739 pre-assigned in the IETF draft for years. While waiting for the IANA registration, we could use this Selector ID.
	5	Reserved.
	6	Reserved.
	7	Reserved.
	8	Reserved.
	9	Reserved.
	10	Reserved.
	11	Reserved.
CANA -L2	12	The Selector ID represents the Cisco Systems unique global layer 2 applications. The Selector ID has a global significance.
CANA -L7	13	The Selector ID represents the Cisco Systems unique global ID for the layer 7 applications. The Selector ID has global significance for all Cisco Systems devices.
	14	Reserved.
	15	Reserved.
	16	Reserved.
	17 to 254	Available.
MAX	255	255 is the maximum Engine ID.

Table 1: Existing Classification Engine IDs

Note 1: "CANA = Cisco Systems Assigned Number Authority", Cisco Systems's version of IANA for internal IDs.

Note 2: This is an extensible list, and new Classification Engine IDs may be allocated at any time. See [CISCO] for the latest version.

4.2. Options Template Record for the Application Name

For engines which specify locally unique Application Tags (which means unique per engine and per router), an Options Template Record (see [RFC5101]) MUST be used to export the correspondence between the Application Tag, the Application Name, and the Application Description. This is called the "options application-table". For engines which specify globally unique Application Tags, an Options Template Record SHOULD be used to export the correspondence between the Application Tag, the Application Name and the Application Description, unless the mapping is hardcoded in the NetFlow Collector, or known out of band (for example, by polling a MIB).

4.3. Resolving IANA L4 port collisions

Even if the IANA L4 ports usually point to the same protocols for both UDP and TCP, there are some exceptions. 10 ports in the first 1K range of ports have different protocols assigned for TCP and UDP:

```
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 machine
login            513/tcp    remote login a la telnet;
#                           automatic authentication performed
#                           based on priviledged port numbers
#                           and distributed data bases which
#                           identify "authentication domains"
who              513/udp    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
  acmaint_transd  775/udp
  busboy          998/tcp
  puparp          998/udp
  garcon          999/tcp
  applix          999/udp    Applix ac
```

Table 2: IANA layer 4 port collisions

Instead of imposing the protocol (UDP/TCP) in the scope of the "options application-table" Options Template for all applications (on top of having the protocol as key-field in the Flow Record definition), we define that the L4 application is always TCP related, by convention. So, whenever the collector has a conflict in looking up IANA, it would choose the TCP choice. The following UDP L4 applications are assigned in the Cisco L7 Application Tag range (ie, under Classification Engine ID 13):

```
comsat/biff     256/udp    used by mail system to notify users
who             257/udp    maintains data bases showing who's
syslog          41/udp
asf-secure-rmcp 258/udp    ASF Secure Remote Management and
#                          Control Protocol
kerberos-iv     259/udp    kerberos version iv
notify          260/udp
acmaint_dbd     261/udp
acmaint_transd  262/udp
puparp          263/udp
applix          264/udp    Applix ac
```

Table 3: Resolving layer 4 UDP ports

5. Application Tag Examples

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

5.1. Example 1: Layer 2 Protocol

From the list of Classification Engine IDs in Table 1, we can see that the layer 2 Classification Engine ID is 12:

```
L2        12    The Selector ID represents the layer 2
                 applications. The Selector ID has a global
                 significance.
```

From the list of layer 2 protocols at [cisco], we can see that PPP has the value 24:

```
NAME    Selector ID
ppp     24
```

So, in the case of layer 2 protocol PPP, the Classification Engine ID is 12 while the Selector ID has the value 24.

Therefore the Application Tag is encoded as:

```
    0                   1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |       12      |      24       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

So the Application Tag has the value of 3097. Instead of representing the Application Tag in hexadecimal format, the format ’12...24’ is used for simplicity in the examples below.

Flexible NetFlow creates a Template Record with a few Information Elements: amongst other things, the Application Tag. For example:

- sourceIPv4Address (key field)
- destinationIPv4Address (key field)
- ipDiffServCodePoint (key field)
- applicationTag (key field)

- octetTotalCount (non key field)

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

```
{ sourceIPv4Address=1.1.1.1, destinationIPv4Address=2.2.2.2,
  ipDiffServCodePoint=0, applicationTag='12...24',
  octetTotalCount=123456 }
```

The Collector has all the required information to determine that the application is PPP, because the Application Tag uses a global and well know registry, ie the IANA protocol number. The 24 value is globally unique within Cisco Systems for Classification Engine ID 12, so the Collector can determine which application is represented by the Application Tag by loading the registry out of band.

5.2. Example 2: Standardized IANA Layer 3 Protocol

From the list of Classification Engine IDs in Table 1, we can see that the IANA layer 3 Classification Engine ID is 1:

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

From the list of IANA layer 3 protocols (see [IANA-PROTO]), we can see that 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 Tag 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 Tag has the value of 257. Instead of representing the Application Tag in hexadecimal format, the format '1...1' is used for simplicity in the examples below.

Flexible NetFlow creates a Template Record with a few Information Elements: amongst other things, the Application Tag. For example:

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

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

```
{ sourceIPv4Address=1.1.1.1, destinationIPv4Address=2.2.2.2,
  ipDiffServCodePoint=0, applicationTag='1...1',
  octetTotalCount=123456 }
```

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

5.3. Example 3: Cisco Systems Proprietary Layer 3 Protocol

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

From the list of Classification Engine IDs in Table 1, we can see that the Cisco Systems layer 3 Classification Engine ID is 2:

CANA-L3	2	Cisco Systems proprietary layer 3 definition. Cisco Systems can still export its own layer 3 protocol numbers, while waiting for IANA to assign it. The Selector ID has a global significance for all Cisco Systems devices under CANA governance. Hopefully the same IDs will be

maintained after the IANA standardization.

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

Protocol	Protocol Id
foo	90

So in the case of Cisco Systems layer 3 protocol foo, the Classification Engine ID is 2, and the Selector ID has the value of 90.

Therefore the Application Tag 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 Tag has the value of 602. Instead of representing the Application Tag in hexadecimal format, the format '2..90' is used for simplicity in the examples below.

Flexible NetFlow creates a Template Record with a few Information Elements: amongst other things, the Application Tag. For example:

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

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

```
{ sourceIPv4Address=1.1.1.1, destinationIPv4Address=2.2.2.2,
  ipDiffServCodePoint=0, applicationTag='2...90',
  octetTotalCount=123456 }
```

Along with this Flow Record, a new Options Template Record would be exported, as shown in Section 5.7.

5.4. Example 4: Standardized IANA Layer 4 Port

From the list of Classification Engine IDs in Table 1, we can see that the IANA layer 4 Classification Engine ID is 3:

```
IANA-       3      IANA layer 4 well-known port number is
 L4                exported in the selector ID.
                   See http://www.iana.org/assignments/port-
                   numbers.

                   Note: as a flow is unidirectional, it
                   contains the destination port in a flow
                   from the client to the server.
```

From the list of IANA layer 4 ports (see [IANA-PORTS]), we can see that 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 Tag is encoded as:

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

Flexible NetFlow creates a Template Record with a few Information Elements: amongst other things, the Application Tag. For example:

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

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

```
{ sourceIPv4Address=1.1.1.1, destinationIPv4Address=2.2.2.2,
  protocol=17, ipDiffServCodePoint=0,
  applicationTag='3..161', octetTotalCount=123456 }
```

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

5.5. Example 4: Layer 7 Application

In this example, the Metering Process has observes some Citrix traffic.

From the list of Classification Engine IDs in Table 1, we can see that the L7 unique Engine ID is 13:

```
 L7        13    The Selector ID represents the Cisco Systems
                 unique global ID for the layer 7
                 application. The Selector ID has a global
                 significance for all Cisco Systems devices.
```

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

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

Therefore the Application Tag 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      13       |                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             10000                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

So the Application Tag has the value of '13..10000'.

Note that the figure shows that the Exporting Process exports the value 10000 in 7 bytes: this is pure speculation. However, it doesn’t matter as the applicationTag would be exported in a variable length Information Element.

Flexible NetFlow creates a Template Record with a few Information Elements: amongst other things, the Application Tag. For example:

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

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

```
{ sourceIPv4Address=1.1.1.1, destinationIPv4Address=2.2.2.2,
  ipDiffServCodePoint=0, applicationTag='13...10000',
  octetTotalCount=123456 }
```

The 10000 value is globally unique within Cisco Systems, so the Collector can determine which application is represented by the Application Tag by loading the registry out of band.

Along with this Flow Record, a new Options Template Record would be exported, as shown in Section 5.7.

5.6. Example: port Obfuscation

For example, a HTTP server might run 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 Tag representation must contain HTTP. However, if the reporting application wants to determine whether or the default HTTP port 80 or 8080 was used, it must export the destination port (destinationTransportPort at [IANA-IPFIX]) in the corresponding NetFlow record.

In the case of a standardized IANA layer 4 port, the Classification Engine ID is 2, and the Selector ID has the value of 80 for HTTP (see [IANA-PORTS]).

Therefore the Application Tag 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               |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Flexible NetFlow creates a Template Record with a few Information Elements: amongst other things, the Application Tag. For example:

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

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

```
    { sourceIPv4Address=1.1.1.1, destinationIPv4Address=2.2.2.2,
      protocol=17, destinationTransportPort=23,
      applicationTag='3..80', octetTotalCount=123456 }
```

The Collector has all the required information to determine that the application is HTTP, but runs on port 23.

5.7. Example: Application Mapping Options Template

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

The Options Template Record would contain the following Information Elements:

1. Scope = applicationTag.

    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=applicationTag='2...90',
  applicationName="foo",
  applicationDescription="The Cisco foo protocol",

  scope=applicationTag='13...10000',
  applicationName="Citrix",
  applicationDescription="A Citrix application" }
```

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

1. A flow of 123456 bytes exists from sourceIPv4Address 1.1.1.1 to destinationIPv4address 2.2.2.2 with a DSCP value of 0 and an applicationTag of '12...90', which maps to the "foo" application.

2. A flow of 123456 bytes exists from sourceIPv4Address 1.1.1.1 to destinationIPv4address 2.2.2.2 with a DSCP value of 0 and an Application Tag of '13...10000', which maps to the "Citrix" application.

6. IANA Considerations

This document specifies three new IPFIX Information Elements: the applicationDescription, applicationTag and the applicationName.

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

EDITOR'S NOTE: the XML specification in Appendix A must be updated with the elementID values allocated below.

6.1. applicationDescription

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

6.2. applicationTag

Name: applicationTag
Description:
  Specifies an Application Tag.
  (EDITOR'S NOTE: reference this document).
Abstract Data Type: octetArray
Data Type Semantics: identifer
ElementId: 95
Status: current

6.3. applicationName

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

7. Security Considerations

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

8. References

8.1. 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.

8.2. 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.

[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.

[IANA-IPFIX] http://www.iana.org/assignments/ipfix/ipfix.xhtml

[IANA-PORTS] http://www.iana.org/assignments/port-numbers

[IANA-PROTO] http://www.iana.org/assignments/protocol-numbers

[CISCO] http://www.cisco.com

9. Acknowledgement

The authors would like to thank their many colleagues across Cisco Systems who made this work possible.

10. 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 (Scotland) Ltd.
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

This appendix 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 Error! Reference source not found.(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="applicationTag"
         dataType="octetArray"
         group="application"
         dataTypeSemantics="identifer"
         elementId="95" applicability="all" status="current">
    <description>
      <paragraph>
         Specifies an Application Tag.
      </paragraph>
    </description>
  </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>
```

IPFIX Working Group
Internet-Draft
Intended Status: Standards Track
Expires: April 25, 2011

B. Claise
Cisco Systems, Inc.
A. Kobayashi
NTT PF Lab.
B. Trammell
Hitachi Europe

October 25, 2010

Specification of the Protocol for IPFIX Mediations
draft-claise-ipfix-mediation-protocol-02

Abstract

This document specifies the IP Flow Information Export (IPFIX) protocol specific to the Mediation.

Status of this Memo

Copyright Notice

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
   1.1. IPFIX Documents Overview..............................4
   1.2. IPFIX Mediator Documents Overview.....................4
   1.3. Relationship with IPFIX and PSAMP.....................5
2. Terminology...6
3. Specifications..9
   3.1. Encoding of IPFIX Message Header......................9
   3.2. Template Management..................................11
      3.2.1. Template Management Without Template Record Change
            11
      3.2.2. Template Management With Template Record Change..12
   3.3. Time Management......................................12
   3.4. Observation Point Management.........................13
      3.4.1. Observation Domain Management...................15
   3.5. Specific Reporting Requirements......................15
      3.5.1. The Flow Keys Options Template..................16
      3.5.2. IPFIX Protocol Options Template.................16
      3.5.3. IPFIX Mediator Options Template.................17
   3.6. The Collecting Process's Side........................17
   3.7. Sampling Management..................................18
4. New Information Elements..................................18
5. Security Considerations...................................18
   5.1. Avoiding Security Downgrade..........................19
   5.2. End-to-End Assertions for Mediators..................19
6. IANA Considerations.......................................20
   6.1. originalExporterIPv4Address..........................21
   6.2. originalExporterIPv6Address..........................21
7. References..21
   7.1. Normative References.................................21
   7.2. Informative References...............................22

8. Author's Addresses...23
9. Appendix A. Additions to XML Specification of IPFIX Information Elements...24

TO DO

- See the EDITOR'S NOTE within the document
- Check that all documents: this one, [IPFIX-MED-ANON], [IPFIX-MED-FLOWSEL], [IPFIX-MED-AGGR] all fit together.

1. Introduction

The IPFIX architectural components in [RFC5470] consist of IPFIX Devices and IPFIX Collectors communicating using the IPFIX protocol [RFC5101], 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 [RFC5102], registered with IANA, or specified as an enterprise-specific Information Element. The specifications in the IPFIX protocol [RFC5101] 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 the Figure A of [IPFIX-MED-FMWK], 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 an Exporting Process and a Collecting 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 the [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 topology, and IPFIX Mediation interpretation, and considerations for aggregation.

The specifications in this document are based on the IPFIX protocol specifications but adapted according to the IPFIX Mediation Framework [IPFIX-MED-FMWK].

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].

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" [IPFIX-MED-FMWK]. 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" [IPFIX-MED-FMWK] 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 Anonymisation Support" [IPFIX-MED-ANON]). Documents specifying the operations of specific Intermediate Processes cover the operation of these Processes within the Mediator framework, and complying to 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 Anonymisation Support", [IPFIX-MED-ANON], which describes anonymisation techniques for IP flow data and the export of anonymised data using the IPFIX protocol.

2. "Flow Selection Techniques" [IPFIX-MED-FLOWSEL], which described the process of selecting a subset of flows from all flows observed at an observation point, along with the motivations, and some specific flow selection techniques.

3. "Exporting Aggregated Flow Data using the IP Flow Information Export" [IPFIX-MED-AGGR] which describes Aggregated Flow export within the framework of IPFIX Mediators and defines an interoperable, implementation-independent method for Aggregated Flow export.

1.3. Relationship with IPFIX and PSAMP

The specification in this document applies to the IPFIX protocol specifications [RFC5101]. All specifications from [RFC5101] 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

The IPFIX-specific and PSAMP-specific terminology used in this document is defined in [RFC5101] and [RFC5476], respectively. The IPFIX Mediation terms related to the aggregation, such as the Interval, Aggregated Flow, and Aggregated Fonction are defined in [IPFIX-MED-AGGR].

The IPFIX Mediation-specific terminology used in this document is defined in "IPFIX Mediation: Problem Statement" [RFC5982], and reuse in "IPFIX Mediation: Framework" [IPFIX-MED-FMWK]. However, since those two documents are an informational RFC, the definitions have been reproduced here along with additional definitions.

Similarly, since the [IPFIX-MED-ANON] is an experimental RFC, the Anonymisation Record, Anonymised Data Record, and Intermediate Anonymisation Process terms, specified in [IPFIX-MED-ANON], are also reproduced here.

In this document, as in [RFC5101], [RFC5476], [IPFIX-MED-AGGR , and [IPFIX-MED-ANON], 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 "record stream" a stream of records carrying flow- or packet-based information. The records may be encoded as IPFIX Data Records in any other format.

Transport Session Information

The Transport Session is specified in [RFC5101]. 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 is a location in the network where IP packets are observed, as received by the IPFIX Mediation. Examples include: 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.

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.

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 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).

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.

Anonymisation Record

A record, defined by the Anonymisation Options Template in section Section 6.1, that defines the properties of the anonymisation applied to a single Information Element within a single Template or Options Template.

Anonymised Data Record

A Data Record within a Data Set containing at least one Information Element with anonymised values. The Information Element(s) within the Template or Options Template describing this Data Record SHOULD have a corresponding Anonymisation Record.

Intermediate Anonymisation Process

An intermediate process which takes Data Records and and transforms them into Anonymised Data Records.

3. Specifications

This section describes the IPFIX specifications for Mediation: more specifically, specifications for generic Intermediate Processes. Possible specific Intermediate Processes are: Intermediate Conversion Process, Intermediate Aggregation Process, Intermediate Correlation Process, Intermediate Selection Process, Intermediate Anonymization Process.

For a specific Intermediate Process, the specifications in the following reference MUST be followed, on the top of the specifications in this document:

- For the Intermediate Aggregation Process, the specifications in [IPFIX-MED-AGGR] MUST be followed.
- For the Intermediate Selection Process, the specifications in [IPFIX-MED-FLOWSEL] MUST be followed.
  EDITOR’S NOTE: actually, there is no MUST/SHOULD/MAY in [IPFIX-MED-FLOWSEL], which seems to be a list of required Information Elements.
- For the Intermediate Anonymization Process, the specifications in [IPFIX-MED-ANON] should be considered as guidelines as [IPFIX-MED-ANON] is an experimental RFC.

Note that no specific document deals with the Intermediate Conversion Process at the time of this publication.

These new specifications, which are more specific compared to [RFC5101], are described with the key words described in [RFC2119].

3.1. Encoding of IPFIX Message Header

The format of the IPFIX Message Header is shown in Figure A. Note that the format is similar to the IPFIX Message in [RFC5101], but some field definitions (for the example, the Export Time) have been updated in the context of the IPFIX Mediator.

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |       Version Number          |            Length             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                           Export Time                         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                       Sequence Number                         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                    Observation Domain ID                      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure A: 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 in seconds since 0000 UTC Jan 1st 1970, at which the IPFIX Message Header leaves the IPFIX Mediator.

Sequence Number

Incremental sequence counter modulo 2^32 of all IPFIX Data Records sent on this PR-SCTP stream from the current Observation Domain by the Exporting Process. Check the specific meaning of this field in the sub-sections 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 is also 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.
Note: the Observation Domain Management is discussed in section 3.4.1.

3.2. Template Management

3.2.1. Template Management Without Template Record Change

The first case is a situation where the IPFIX Mediator, typically an IPFIX Distributor, relays an (Options) Template without changing its content.

As in [RFC5101], 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 mapping database between received and exported (Options) Template Records:

- for each Received (Options) Template Record: Template Record Flow Keys and non Flow Keys, Template ID, Original Exporter, Observation Domain, and Transport Session
- for each Exported (Options) Template Record: Template Record Flow Keys and non Flow Keys, Template ID, Collector, Observation Domain, and Transport Session

If an IPFIX Mediator receives an IPFIX Withdrawal Message for a (Options) Template Record that is not used anymore in any outgoing Transport Sessions, the IPFIX Mediator SHOULD export the appropriate IPFIX Withdrawal Message(s) on the outgoing Transport Session, and remove the corresponding entry in its mapping database.

If a (Options) Template Record is not used anymore in outgoing Transport Session, it MUST be withdrawn with an IPFIX Template Withdrawal Message on that specific outgoing Transport Session, and remove the corresponding entry in its mapping database.

If an incoming Transport Session is gracefully shutdown or reset, the (Options) Template Records corresponding to that Transport Session MUST be removed from the mapping database.

3.2.2. Template Management With Template Record Change

The second case is a situation where the IPFIX Mediator, typically containing an Intermediate Conversion Process, Intermediate Aggregation Process [IPFIX-MED-AGGR], or Intermediate Anonymization Process in case of black-marker anonymisation [IPFIX-MED-ANON], generates new (Options) Template Records based what it receives from the Original Exporter(s), and based on the Intermediate Process function.

In such a situation, the IPFIX Mediator doesn’t maintain a mapping database between received and exported (Options) Template Records, as it generates its own series of (Options) Template Records.

3.3. Time Management

The IPFIX Message Header "Export Time" field is the time in seconds since 0000 UTC Jan 1, 1970, at which the IPFIX Message Header 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 [RFC5102], 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 Intermediate Processes: Intermediate Aggregation Process (temporal) and Intermediate Correlation Process, for example.

When an Intermediate Aggregation Process aggregates information from different Flow Records, the typical reporting times SHOULD BE the minimum of the start times and the maximum of the end times. 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, the precise rules of applying the Flow Record timestamps in IPFIX Mediators is out of the scope of this document. However, some more specifications related to the specific case of aggregation in space and time are specified in [IPFIX-MED-AGGR], and MUST be followed.

3.4. Observation Point Management

Depending on the use case, top Collectors may need to receive the Original Observation Point(s), otherwise it may wrongly conclude that the IPFIX Device exporting the Flow Records to him, i.e. the IPFIX Mediator, directly observed the packets that generated the Flow Records. Two new Information Element are introduced to solve this use case: originalExporterIPv4Address and originalExporterIPv6Address.

In the IPFIX Mediator, the Observation Point(s) may be represented by:

- A single Original Exporter (represented by the originalExporterIPv4Address or originalExporterIPv6Address Information Elements)
- A list of Original Exporter (represented by the originalExporterIPv4Address or originalExporterIPv6Address Information Elements_)
- A list of Original Exporter (represented by the originalExporterIPv4Address or originalExporterIPv6Address Information Elements), along with the associated interface (represented by the ingressInterface and/or egressInterface)
- A list of Original Exporter (represented by the originalExporterIPv4Address or originalExporterIPv6Address Information Elements), along with the associated line card id (represented by the lineCardId)
- Any combination or list of Information Elements representing Observation Points.

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 examples is possible. For example, a Original Observation Point 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
```

If the Original Observation Point composed of a list exported from the IPFIX Mediator, then the IPFIX Structured Data [IPFIX-STRUCT] MUST be used to encode it.

The most generic way to export the Original Observation Point is to use a subTemplateMultiList, with the semantic "exactlyOneOf". Taking back the previous example, the following encoding can be used:

```
         Template Record 1: exporterIPv4Address
         Template Record 2: exporterIPv4Address, basicList of
                            ingressInterface, flowDirecdtion
```

Template Record 3: exporterIPv4Address, lineCardId, flowDirection

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 exports the Original Observation Domain, it SHOULD export other information indicating that an IPFIX Mediator certificates the original exporter IP address. ExporterCertificate in [RFC5655] can be used in that case. And also, another Information Element indicating that certifies that an IPFIX Mediator is required, just like mediatorCertificate.

3.4.1. Observation Domain Management

In terms of Observation Domain management, there are two types of Intermediate Process function. The first one maintains the Observation Domain information, while the second one cannot maintain it. Examples of the second type include mixing Data Records from multiple IPFIX Messages received from multiple Observation Domains, or generating new Data Records from the result of some intermediate function on Data Records from multiple IPFIX Messages received from multiple Observation Domains.

From the two types of Intermediate Process function discussed in this section, a single specification can be deduced, as already specified in 3.1. : The Observation Domain ID SHOULD be 0 when no specific Observation Domain ID is relevant for the entire IPFIX Message.

3.5. 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 Records defined in these subsections, which impose some constraints on the Metering Process and Exporting Process implementations in Intermediate Processes, MAY be implemented. If implemented, the specific Option 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.

3.5.1. The Flow Keys Options Template

Exactly like the IPFIX protocol [RFC5101], 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.

The Flow Keys Option Template SHOULD contain the following Information Elements that are defined in [RFC5102]

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.

When any Intermediate Process changes the Flow Keys, the Flow Keys Option Template MUST include the new set of Flow Keys. Typically, an Intermediate Aggregation Process keeps or reduces the number of Flow Keys

3.5.2. IPFIX Protocol Options Template

The "Metering Process Statistics Options Template", "The Metering Process Reliability Statistics Options Template", and "The Exporting Process Reliability Statistics Options Template", as specified in [RFC5101], SHOULD be implemented on the IFPIX Mediator.

Refer to the document specifying a particular Intermediate Process type for specific values for these Options Template Records. For example, in case of an Intermediate Aggregation Process, [IPFIX-MED-AGGR] must specify which values to insert into the fields of "Metering Process Statistics Options Template", "The Metering Process Reliability Statistics Options

Template", and "The Exporting Process Reliability Statistics Options Template"

3.5.3. IPFIX Mediator Options Template

There is no need for a specific Options Template for the IPFIX Mediator; instead, each Intermediate Process type requires some particular metadata. For example, a specification of IPFIX flow anonymisation including an Options Template for the export of metadata about anonymised flows is described in [IPFIX-MED-ANON]; when anonymising Flows Records, IPFIX Mediators SHOULD add the Options Template specified therein to annotate the exported dataTransport Session Management
SCTP [RFC4960] using the PR-SCTP extension specified in [RFC3758] MUST be implemented by all compliant IPFIX Mediator implementations. UDP [UDP] MAY also be implemented by compliant IPFIX Mediator implementations. TCP [TCP] MAY also be implemented by IPFIX Mediator compliant implementations.

PR-SCTP SHOULD be used in deployments where IPFIX Mediators 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 IPFIX Mediators and Collectors communicate over links that are susceptible to congestion, but PR-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, 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.

3.6. The Collecting Process’s Side

An IPFIX Mediator MUST produce IPFIX Messages understandable by a RFC5101-compliant IPFIX Collector, with the additional specification in the IPFIX Structured Data [IPFIX-STRUCT].

Therefore the Collecting Process on the top Collector MUST support the IPFIX protocol [RFC5101] and the IPFIX Structured Data [IPFIX-STRUCT].

3.7. Sampling Management

EDITOR'S NOTE: What about the accuracy of aggregated Flow Records with the sampling rates? With different sampling rates?

In an IPFIX Mediation, aggregation for Flow Records with same sampling rate and same sampling algorithm is recommended. In that case, an IPFIX Mediator can export this sampling rate and sampling algorithm, and other accuracy statistics data, part of the PSAMP Report Interpretation [RFC5476].

In the case where the Mediation aggregates Flow Records with different sampling functions and/or sampling rates, some more research is required to determine the right sampling function and/or sampling rate to export from the IPFIX Mediator. Therefore, this document doesn't describe any specifications, or even guidelines.

4. New Information Elements

- originalExporterIPv4Address and originalExporterIPv6Address

EDITOR'S NOTE: to be discussed

- orginalObservationDomainId?
- mediatorCertificate?

EDITOR'S NOTE: Maybe the following ones should be defined in a specific flow aggregation draft:

- Maximum counter or minimum counter for packets or bytes
- activeTime and inactiveTime for Flow aggregation

5. Security Considerations

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

As they act as both IPFIX Collecting Processes and Exporting Processes, the Security Considerations for IPFIX [RFC5101] apply as well to Mediators. The Security Considerations for IPFIX Files [RFC5655] apply as well 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 take into account in addition.

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. We address these in the following subsections.

5.1. Avoiding Security Downgrade

An IPFIX Mediator that accepts IPFIX Messages over a Transport Session protected by TLS or DTLS, and which then exports IPFIX Messages derived there from in cleartext, is a potentially serious vulnerability in an IPFIX infrastructure. While this is potentially acceptable in the specific case of an IPFIX Mediator at the border of an administrative domain accepting IPFIX Messages from outside the domain and re-exporting derived information via an internal network protected by other means, in the general case this situation SHOULD be avoided.

Therefore, an IPFIX Mediator that receives IPFIX Messages from an upstream Exporting Process protected using TLS or DTLS MUST provide for sending of IPFIX Messages resulting from the Intermediate Process to a downstream Collecting Process using TLS or DTLS. It MAY allow for the configuration of unprotected export of such IPFIX Messages, but in this case it MUST warn the administrator that the exported IPFIX Messages will not be protected, and that this could result in the leakage of information deemed by the Original Exporter to be worth protecting.

5.2. End-to-End Assertions for Mediators

Because the Transport Session between an IPFIX Mediator and an Original Exporter is independent from the Transport Session between the Mediator and the downstream Collecting Process, there exists no method via TLS to assert the identity of the original Exporting Process downstream. However, an IPFIX Mediator, which modifies the stream of IPFIX Messages sent to it, is by definition a trusted entity in the infrastructure. Therefore, the IPFIX Mediator’s signature on an outgoing Transport Session can be treated as an implicit assertion that the Original Exporter was positively identified by the Mediator and that the source information it received was trustworthy.

However, IPFIX Mediators must in this circumstance take care not to provide an inappropriate upgrade of trust.

Therefore, an IPFIX Mediator SHOULD NOT sign a Transport Session to a downstream Collector unless ALL the Original Exporters from which the information to be exported is derived were positively identified by the Mediator by its certificate. An exception to this case is the reverse of the special case in the previous subsection: an IPFIX Mediator that accepts information from within a trusted domain via an internal network protected by other means MAY use TLS or DTLS to protect the Transport Session to a downstream Collector outside the domain.

[EDITOR OPEN ISSUE: We might want to use exporterCertificate and (optionally) collectorCertificate from [RFC5655] here, but I think they need a new Mediator-specific template if so. If we were to use the templates defined by 5655, it would look like this:

If the X.509 certificates used to protect a Transport Session between an Original Exporter and an IPFIX Mediator are required downstream, an IPFIX Mediator MAY use the exporterCertificate and the collectorCertificate Information Elements with the Export Session Details Options Template defined in Section 8.1.3 of [RFC5655] or the Message Details Options Template defined in Section 8.1.4. of [RFC5655] in order to export this information downstream. However, in this case, the IPFIX Mediator is making an implicit assertion that the upstream Session was properly protected and therefore trustworthy, and as such MUST protect the Transport Session to the downstream Collector using TLS or DTLS, as well.

6. IANA Considerations

This document specifies three new IPFIX Information Elements: the applicationDescription, applicationTag and the applicationName.

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

EDITOR'S NOTE: the XML specification in Appendix A must be updated with the elementID values allocated below.

6.1. originalExporterIPv4Address

Name: originalExporterIPv4Address
Description:
The IPv4 address used by the Exporting Process on the Original Observation Point. This is used by the Exporting Process (on the Mediation) to identify the Exporter in cases where the identity of the Exporter may have been obscured by the use of a proxy, or in cases where the IPFIX Mediation must export the Original Observation Point to a top Collector.
Abstract Data Type: ipv4Address
Data Type Semantics: identifier
ElementId: XXX
Status: current

6.2. originalExporterIPv6Address

Name: originalExporterIPv6Address
Description:
The IPv6 address used by the Exporting Process on the Original Observation Point. This is used by the Exporting Process (on the Mediation) to identify the Exporter in cases where the identity of the Exporter may have been obscured by the use of a proxy, or in cases where the IPFIX Mediation must export the Original Observation Point to a top Collector.
Abstract Data Type: ipv6Address
Data Type Semantics: identifier
ElementId: YYY
Status: current

EDITOR’S NOTE: maybe some more IEs

7. References

7.1. Normative References

[RFC2119] S. Bradner, 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", May 2004

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

[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.

[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.

[IPFIX-MED-FLOWSEL] Peluso, L., Zseby, T., D'antonio, S., and M. Molina, "Flow Selection Techniques", draft-ietf-ipfix-flow-selection-tech-02.txt, Internet-Draft work in progress, June 2010.

[IPFIX-MED-AGGR] Trammell, B., Boschi, E., and A. Wagner, "Exporting Aggregated Flow Data using the IP Flow Information Export (IPFIX) Protocol", draft-trammell-ipfix-a9n-01.txt, Internet-Draft work in progress, October 2010.

[IPFIX-STRUCT] Claise, B., Dhandapani, G., Aitken, P., and S. Yates, "Export of Structured Data in IPFIX", draft-ietf-ipfix-structured-data-03.txt, Internet-Draft work in progress, October 2010.

7.2. Informative References

[TCP] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[UDP] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

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

[RFC3954] Claise, B. (Ed), "Cisco Systems NetFlow Services Export Version 9", RFC 3954, October 2004

[RFC5470] Sadasivan, G., Brownlee, N., Claise, B., and J. Quittek, "Architecture Model for IP Flow Information Export", RFC5470, 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., Quittek, J., and A. Johnson, "Packet Sampling (PSAMP) Protocol Specifications", RFC 5476, March 2009.

[RFC5982] Kobayashi, A. (Ed), Claise, B. (Ed), "P Flow Information Export (IPFIX) Mediation: Problem Statement", RFC 5982, August 2010.

[IPFIX-MED-FMWK] Kobayashi, A., Claise, B., and K. Ishibashi, "IPFIX Mediation: Framework", draft-ietf-ipfix-mediators-framework-09, Internet-Draft work in progress, October 2010.

[IPFIX-MED-ANON] Boschi, E., Trammell, B. "IP Flow Anonymisation Support", draft-ietf-ipfix-anon-05.txt, Internet-Draft work in progress, October 2010.

[IANA-IPFIX] http://www.iana.org/assignments/ipfix/ipfix.xhtml

8. Author's Addresses

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

URI: http://www3.plala.or.jp/akoba/

Brian Trammell
Hitachi Europe
c/o ETH Zurich
Gloriastrasse 35
8092 Zurich
Switzerland

Phone: +41 44 632 70 13
EMail: brian.trammell@hitachi-eu.com

9. Appendix A. Additions to XML Specification of IPFIX Information Elements

This appendix 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 6. (generated from this appendix) is normative.

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

```
<field name=" originalExporterIPv4Address "
         dataType="identifier "
         group="config"
         elementId="XXX" applicability="all" status="current">
    <description>
      <paragraph>
        The IPv4 address used by the Exporting Process on the
        Original Observation Point. This is used by the
        Exporting Process (on the Mediation) to identify the
        Exporter in cases where the identity of the Exporter
        may have been obscured by the use of a proxy, or in
        cases where the IPFIX Mediation must export the
        Original Observation Point to a top Collector.
      </paragraph>
    </description>
  </field>


<field name=" originalExporterIPv6Address "
         dataType="identifier "
         group="config"
         elementId="XXX" applicability="all" status="current">
```

```
       <description>
         <paragraph>
           The IPv6 address used by the Exporting Process on the
           Original Observation Point. This is used by the
           Exporting Process (on the Mediation) to identify the
           Exporter in cases where the identity of the Exporter
           may have been obscured by the use of a proxy, or in
           cases where the IPFIX Mediation must export the
           Original Observation Point to a top Collector.
         </paragraph>
       </description>
     </field>
```

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 12, 2011

T. Dietz, Ed.
NEC Europe Ltd.
B. Claise
Cisco Systems, Inc.
J. Quittek
NEC Europe Ltd.

November 8, 2010

Definitions of Managed Objects for Packet Sampling
<draft-ietf-ipfix-psamp-mib-02.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 MIB module [RFC5815]. For IPFIX implementations that use packet sampling (PSAMP) techniques as described in [RFC5475], 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

Copyright Notice

Table of Contents

1. Open Issues/TODOs 3

2. The Internet-Standard Management Framework 3

3. Introduction 3

4. PSAMP Documents Overview 4

5. Related IPFIX Documents 4

6. Structure of the PSAMP MIB module 4
  6.1. Textual Conventions 5
  6.2. Packet Selection Functions 6
    6.2.1. Systematic Count-based Sampling 6
    6.2.2. Systematic Time-based Sampling 6
    6.2.3. Random n-out-of-N Sampling 7
    6.2.4. Uniform Probabilistic Sampling 7
    6.2.5. Property Match Filtering 7
    6.2.6. Hash-based Filtering 8

7. Definitions 8

8. Security Considerations 24

9. IANA Considerations 25

10. Acknowledgment 26

11. References 26
  11.1. Normative References 26
  11.2. Informative References 26

Authors' Addresses 27

1. Open Issues/TODOs

- o security considerations: check security issues raised by Nick Duffield on privacy issues with hash parameters etc.

2. 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 MIB modules 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].

3. Introduction

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].

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 hashing 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 module [RFC5815]. Since the IPFIX MIB module is for monitoring only 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 6 describes the structure of the PSAMP MIB module and section 7 contains the formal definition. Security issues are discussed in section 8.

4. 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.

5. Related IPFIX Documents

The IPFIX protocol provides network administrators with access to IP Flow information. The protocol document [RFC5101] specifies how IPFIX Data Records and Templates are carried via a congestion-aware transport protocol from IPFIX Exporting Processes to IPFIX Collecting Processes. This document also specifies the data types used in this MIB module and their encoding. The IPFIX MIB [RFC5815] is the basis for this document and is extended by this MIB module.

6. Structure of the PSAMP MIB module

The IPFIX MIB module defined in [RFC5815] 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. This table 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 that only contains one more object indicating the current availability of this function. For functions that 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.

Object ipfixSelectorFunctions in the IPFIX SELECTOR MIB module serves as home for objects that describe instances of packet selector functions. The IPFIX SELECTOR MIB is a very small module that is also defined in [RFC5815]. Objects under ipfixSelectorFunctions are maintained by IANA. In the IPFIX SELECTOR module 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 contains six new objects under 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 [RFC5815]. All functions and their parameters are already listed in the overview of functions given by the figure in section 8.2.1 of [RFC5477].

In addition, the PSAMP MIB module contains two textual conventions that define data types used for parameters in the above tables that cannot be expressed by the basic data types defined by [RFC2578], unsigned64 and float64.

6.1. Textual Conventions

The MIB module defines one textual conventions that defines a data type used within this MIB module. Another data type is imported from the APPLICATION MIB [RFC2564]. Those data types are defined according to [RFC5101]. Those data types are not 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.

The PsampFloat64 textuanl convention describes a double precision floating point number. It is encoded accroding to [IEEE.754.1985].

6.2. 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.

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].

6.2.1. 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)
```

6.2.2. 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 previous one. 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)
```

6.2.3. Random n-out-of-N Sampling

The third selector function is random n-out-of-N sampling. The structure of the sub-tree for this function is similar to the previous one. Parameters are psampSampRandOutOfNSamplingSize and psampSampRandOutOfNPopulation:

```
psampSampRandOutOfN(4)
+-- psampSampRandOutOfNAvail(1)
+-- psampSampRandOutOfNParamSetTable(3)
   +-- psampSampRandOutOfNParamSetEntry(1) [psampSampRandOutOfNIndex]
      +-- psampSampRandOutOfNIndex(1)
      +-- psampSampRandOutOfNSamplingSize(2)
      +-- psampSampRandOutOfNPopulation(3)
```

6.2.4. Uniform Probabilistic Sampling

The fourth selector function is uniform probabilistic sampling. It has just a single parameter called psampSampUniProbProbability:

```
psampSampUniProb(5)
+-- psampSampUniProbAvail(1)
+-- psampSampUniProbParamSetTable(3)
   +-- psampSampUniProbParamSetEntry(1) [psampSampUniProbIndex]
      +-- psampSampUniProbIndex(1)
      +-- psampSampUniProbProbability(2)
```

6.2.5. 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. Paameters cannot be retireved via the PSAMP MIB module:

```
psampFiltPropMatch(6)
+-- psampFiltPropMatchAvail(1)
```

6.2.6. Hash-based Filtering

The sixth selector function is hash-based filtering. This function has more 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.

7. Definitions

```
PSAMP-MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, Integer32, Unsigned32, mib-2
        FROM SNMPv2-SMI                        -- RFC2578
    TEXTUAL-CONVENTION, TruthValue
        FROM SNMPv2-TC                         -- RFC2579
    MODULE-COMPLIANCE, OBJECT-GROUP
        FROM SNMPv2-CONF                       -- RFC2580
    Unsigned64TC
        FROM APPLICATION-MIB                   -- RFC2564
    ipfixSelectorFunctions
```

```
           FROM IPFIX-SELECTOR-MIB;

   psampMIB MODULE-IDENTITY
       LAST-UPDATED "201011081200Z"         -- 08 November 2010
       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@nw.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@nw.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.

           Copyright (c) 2010 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     "201011081200Z"         -- 08 November 2010
        DESCRIPTION
            "Initial version, published as RFC yyyy."
   -- RFC Ed.: replace yyyy with actual RFC number & remove this notice

       ::= { mib-2 xxx }
   -- xxx to be assigned by IANA.

   PsampFloat64 ::= TEXTUAL-CONVENTION
       STATUS       current
       DESCRIPTION
               "Represents the float64 data type and MUST be encoded as
               an IEEE double-precision 64-bit floating point-type, as
               specified in IEEE 754."
       SYNTAX       OCTET STRING (SIZE (8))

   -- 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."
       DEFVAL { false }
       ::= { 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 the 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 ipfixSelectionProcessSelectorFunctionentries 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."
      DEFVAL { false }
      ::= { 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 ipfixSelectionProcessSelectorFunctionentries 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."
       DEFVAL { false }
```

```
        ::= { 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 a two parameter only, the sampling size and the
            parent population."
        ::= { psampSampRandOutOfN 3 }

    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,
            psampSampRandOutOfNSamplingSize 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 ipfixSelectionProcessSelectorFunctionentries of
            the ipfixSelectionProcessTable in the IPFIX-MIB as reference
            to this parameter set."
        ::= { psampSampRandOutOfNParamSetEntry 1 }

    psampSampRandOutOfNSamplingSize 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 for 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."
       DEFVAL { false }
       ::= { 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 3 }

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 PsampFloat64
    }

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 ipfixSelectionProcessSelectorFunctionentries of
        the ipfixSelectionProcessTable in the IPFIX-MIB as reference
        to this parameter set."
    ::= { psampSampUniProbParamSetEntry 1 }

psampSampUniProbProbability OBJECT-TYPE
    SYNTAX      PsampFloat64
    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 was sampled since the probability is 0. A value
        of 1 means all packets were sampled since the
        probability is 1."
    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."
     DEFVAL { false }
     ::= { psampFiltPropMatch 1 }

 --=====================================================================
 --* Method 1: 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."
       DEFVAL { false }
       ::= { 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
           parameter 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 ipfixSelectionProcessSelectorFunctionentries 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
           supports the following Hash Functions:

           crc32(1)

           ipsx(2)

           bob(3)
           "
       ::= { 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 the corresponding
               sampling function is implemented in the equipment."
           GROUP psampGroupSampTimeBased
           DESCRIPTION
               "These objects must be implemented if the corresponding
               sampling function is implemented in the equipment."
           GROUP psampGroupSampRandOutOfN
           DESCRIPTION
               "These objects must be implemented if the corresponding
               sampling function is implemented in the equipment."
           GROUP psampGroupSampUniProb
           DESCRIPTION
               "These objects must be implemented if the corresponding
               sampling function is implemented in the equipment."
           GROUP psampGroupFiltPropMatch
           DESCRIPTION
               "These objects must be implemented if the corresponding
               filter function is implemented in the equipment."
           GROUP psampGroupFiltHash
           DESCRIPTION
               "These objects must be implemented if the corresponding
```

```
                filter function 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 2 }

   psampGroupSampTimeBased OBJECT-GROUP
       OBJECTS {
                 psampSampTimeBasedAvail,
                 psampSampTimeBasedInterval,
                 psampSampTimeBasedSpace
               }
       STATUS  current
       DESCRIPTION
          "These objects are needed if time based sampling is
          implemented."
       ::= { psampGroups 3 }

   psampGroupSampRandOutOfN OBJECT-GROUP
       OBJECTS {
                 psampSampRandOutOfNAvail,
                 psampSampRandOutOfNSamplingSize,
                 psampSampRandOutOfNPopulation
               }
       STATUS  current
       DESCRIPTION
          "These objects are needed if random n-out-of-N sampling is
          implemented."
       ::= { psampGroups 4 }

   psampGroupSampUniProb OBJECT-GROUP
       OBJECTS {
                 psampSampUniProbAvail,
                 psampSampUniProbProbability
               }
```

```
        STATUS  current
        DESCRIPTION
           "These objects are needed if uniform probabilistic sampling
           is implemented."
        ::= { psampGroups 5 }

    psampGroupFiltPropMatch OBJECT-GROUP
        OBJECTS {
                  psampFiltPropMatchAvail
                }
        STATUS  current
        DESCRIPTION
           "These objects are needed if property match filtering is
           implemented."
        ::= { psampGroups 6 }

    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 9 }

    END
```

8. 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.

Some of the readable objects in this MIB module (i.e., objects with a MAX-ACCESS other than not-accessible) may be considered sensitive or vulnerable in some network environments. It is thus important to control even GET and/or NOTIFY access to these objects and possibly to even encrypt the values of these objects when sending them over the network via SNMP. These are the tables and objects and their sensitivity/vulnerability:

- o All tables - they contain configuration data that might be sensitive because objects in this table may reveal information about the network infrastructure and device configuration

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 as 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) that have legitimate rights to indeed GET or SET (change/create/delete) them.

9. 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 }
psampSampCountBased    { ipfixSelectorFunctions 2 }
psampSampTimeBased     { ipfixSelectorFunctions 3 }
psampSampRandOutOfN    { ipfixSelectorFunctions 4 }
psampSampUniProb       { ipfixSelectorFunctions 5 }
psampFiltPropMatch     { ipfixSelectorFunctions 6 }
psampFiltHash          { ipfixSelectorFunctions 7 }
```

Other than that this document does not impose any IANA considerations.

10. Acknowledgment

This document is a product of the PSAMP and IPFIX working groups.

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.

[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.

[RFC5815] Dietz, T., Kobayashi, A., Claise, B., and G. Muenz, "Definitions of Managed Objects for IP Flow Information Export", RFC 5815, April 2010.

[IEEE.754.1985] Institute of Electrical and Electronics Engineers, "Standard for Binary Floating-Point Arithmetic", IEEE Standard 754, August 1985.

11.2. 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

IPFIX Working Group
Internet-Draft
Intended Status: Standards Track
Expires: April 10, 2011
B. Claise
G. Dhandapani
P. Aitken
S. Yates
Cisco Systems, Inc.
October 10, 2010

Export of Structured Data in IPFIX
draft-ietf-ipfix-structured-data-03.txt

Status of this Memo

Abstract

This document specifies an extension to the IP Flow Information eXport (IPFIX) protocol specification in [RFC5101] and the IPFIX information model specified in [RFC5102] to support hierarchical structured data and lists (sequences) of Information Elements in data records. This extension allows definition of complex data structures such as variable-length lists and specification of hierarchical containment relationships between Templates. Finally, the semantics are provided in order to express the relationship among multiple list elements in a structured data record.

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. Overview.. 7
   1.1. IPFIX Documents Overview........................... 7
   1.2. Relationship between IPFIX and PSAMP............... 8
2. Introduction.. 8
   2.1. The IPFIX Track.................................... 9
   2.2. The IPFIX Limitations............................. 10
   2.3. Structured Data Use Cases......................... 10
   2.4. The Proposal...................................... 12
3. Terminology.. 13
   3.1. New Terminology................................... 13
4. Linkage with the IPFIX Information Model............... 13
   4.1. New Abstract Data Types........................... 14
      4.1.1. basicList.................................... 14
      4.1.2. subTemplateList.............................. 14
      4.1.3. subTemplateMultiList......................... 14
   4.2. New Data Type Semantic............................ 14
      4.2.1. List... 15
   4.3. New Information Elements.......................... 15
      4.3.1. basicList.................................... 15
      4.3.2. subTemplateList.............................. 15
      4.3.3. subTemplateMultiList......................... 15
   4.4. New Structured Data Type Semantics................ 16
      4.4.1. undefined.................................... 16
      4.4.2. noneOf....................................... 16
      4.4.3. exactlyOneOf................................. 16
      4.4.4. oneOrMoreOf.................................. 17
      4.4.5. allOf.. 18
      4.4.6. ordered...................................... 18
   4.5. Encoding of IPFIX Data Types...................... 19
      4.5.1. basicList.................................... 19
      4.5.2. subTemplateList.............................. 21
      4.5.3. subTemplateMultiList......................... 23
5. Structured Data Format................................. 26
   5.1. Length Encoding Considerations.................... 26
   5.2. Recursive Structured Data......................... 27
   5.3. Structured Data Information Elements Applicability in
   Options Template Sets.................................. 27
   5.4. Usage Guidelines for Equivalent Data Representations28
   5.5. Padding... 29
   5.6. Semantic.. 29
6. Template Management.................................... 33
7. The Collecting Process’s Side.......................... 34
8. Structured Data Encoding Examples...................... 34
   8.1. Encoding a Multicast Data Record with basicList.... 34

8.2. Encoding a Load-balanced Data Record with a basicList36
8.3. Encoding subTemplateList.......................... 37
8.4. Encoding subTemplateMultiList..................... 40
8.5. Encoding an Options Template Set using Structured Data.. 45
9. Relationship with the Other IFPIX Documents............. 49
9.1. Relationship with Reducing Redundancy.............. 49
9.1.1. Encoding Structured Data Element using Common Properties... 50
9.1.2. Encoding Common Properties elements With Structured Data Information Element.................. 50
9.2. Relationship with Guidelines for IPFIX Testing..... 52
9.3. Relationship with Bidirectional Flow Export........ 53
9.4. Relationship with IPFIX Mediation Function......... 54
10. IANA Considerations.................................... 54
10.1. New Abstract Data Types.......................... 54
10.1.1. basicList..................................... 54
10.1.2. subTemplateList............................... 54
10.1.3. subTemplateMultiList.......................... 55
10.2. New Data Type Semantics.......................... 55
10.2.1. list.. 55
10.3. New Information Elements......................... 55
10.3.1. basicList..................................... 55
10.3.2. subTemplateList............................... 56
10.3.3. subTemplateMultiList.......................... 56
10.4. New Structured Data Semantics.................... 56
10.4.1. undefined..................................... 56
10.4.2. noneOf.. 57
10.4.3. exactlyOneOf.................................. 57
10.4.4. oneOrMoreOf................................... 57
10.4.5. allOf... 57
10.4.6. ordered....................................... 58
11. Security Considerations................................ 58
12. References... 58
12.1. Normative References............................. 58
12.2. Informative References........................... 59
13. Acknowledgement.. 59
14. Authors' Addresses..................................... 60
Appendix A. Additions to XML Specification of IPFIX Information Elements and Abstract Data Types............... 61
Appendix B. Example of Biflow Encoding using Structured Data Information Elements..................................... 66
Appendix C. Encoding IPS Alert using Structured Data Information Elements..................................... 68

Table of Figures

Figure A: basicList Encoding................................ 19
Figure B: basicList Encoding with Enterprise Number.......... 20
Figure C: Variable-Length basicList Encoding (Length < 255 octets) 21
Figure D: Variable-Length basicList Encoding (Length 0 to 65535 octets) 21
Figure E: subTemplateList Encoding.......................... 22
Figure F: Variable-Length subTemplateList Encoding (Length < 255 octets) 23
Figure G: Variable-Length subTemplateList Encoding (Length 0 to 65535 octets) 23
Figure H: subTemplateMultiList Encoding...................... 24
Figure I: Variable-Length subTemplateMultiList Encoding (Length < 255 octets) 25
Figure J: Variable-Length subTemplateMultiList Encoding (Length 0 to 65535 octets) 26
Figure K: Encoding basicList, Template Record................ 35
Figure L: Encoding basicList, Data Record, Semantic allOf.... 36
Figure M: Encoding basicList, Data Record with Variable-Length Elements, Semantic allOf 36
Figure N: Encoding basicList, Data Record, Semantic ExactlyOneOf 37
Figure O: Encoding subTemplateList, Template for One-Way Delay Metrics 38
Figure P: Encoding subTemplateList, Template Record.......... 39
Figure Q: Encoding subTemplateList, Data Set................. 40
Figure R: Encoding subTemplateMultiList, Template for Filtering Attributes 43
Figure S: Encoding subTemplateMultiList, Template for Sampling Attributes 43
Figure T: Encoding subTemplateMultiList, Template for Flow Record 44
Figure U: Encoding subTemplateMultiList, Data Set............ 45
Note that the example could further be improved with a basicList of selectorId if many Selector IDs have to be reported. .. 47
Figure V: PSAMP SSRI to be encoded.......................... 47
Figure W: Options Template Record for PSAMP SSRI using subTemplateMultiList 47
Figure X: PSAMP SSRI, Template Record for interface.......... 48
Figure Y: PSAMP SSRI, Template Record for linecard........... 48
Figure Z: PSAMP SSRI, Template Record for linecard and interface 48
Figure ZA: Example of a PSAMP SSRI Data Record, Encoded using a subTemplateMultiList 49

Figure ZB: Common and Specific Properties Exported Together
   [RFC5473] 51
Figure ZC: Common and Specific Properties Exported Separately
   according to [RFC5473] 51
Figure ZD: Common and Specific Properties Exported with
   Structured Data Information Element 51
Figure B0: Using a subTemplateList to represent a Biflow..... 66
Figure B1: Template for the Biflow Fields.................... 67
Figure B2: Template for the Key Fields....................... 67
Figure B3: Biflow Data Set Encoded using Structured Data..... 68
Figure C0: Encoding IPS Alert, Template for Target........... 71
Figure C1: Encoding IPS Alert, Template for Attacker......... 71
Figure C2: Encoding IPS Alert, Template for Participant...... 72
Figure C3: Encoding IPS Alert, Template for IPS Alert........ 72
Figure C4: Encoding IPS Alert, Data Set...................... 74

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 [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.

1.2. Relationship between IPFIX and PSAMP

The specification in this document applies to the IPFIX protocol specifications [RFC5101]. All specifications from [RFC5101] apply unless specified otherwise in this document.

The Packet Sampling (PSAMP) protocol [RFC5476] specifies the export of packet information from a PSAMP Exporting Process to a PSAMP Collecting Process. Like IPFIX, PSAMP has a formal description of its information elements, their name, type and additional semantic information. The PSAMP information model is defined in [RFC5477].

As the PSAMP protocol specifications [RFC5476] are based on the IPFIX protocol specifications, the specifications in this document are also valid for the PSAMP protocol.

Indeed, the major difference between IPFIX and PSAMP is that the IPFIX protocol exports Flow Records while the PSAMP protocol exports Packet Reports. From a pure export point of view, IPFIX will not distinguish a Flow Record composed of several packets aggregated together, from a Flow Record composed of a single packet. So the PSAMP export can be seen as a special IPFIX Flow Record containing information about a single packet.

2. Introduction

While collecting the interface counters every five minutes has proven to be useful in the past, more and more granular information is required from network elements for a series of applications: performance assurance, capacity planning, security, billing, or simply monitoring. However, the amount of information has become so important that, when dealing with highly granular information such as Flow information, a push mechanism (as opposed to a pull mechanism, such as SNMP) is the only solution for routers whose primary function is to route packets. Indeed, polling short-live Flows via SNMP is not an option: high end routers can support hundreds of thousands of Flows simultaneously. Furthermore, in order to reduce the export bandwidth requirements, the network elements have to integrate mediation functions to aggregate the collected information, both in space and time.

Typically, it would be beneficial if access routers could export Flow Records, composed of the counters before and after an optimization mechanism on the egress interface, instead of exporting two Flow Records with identical tuple information.

In terms of aggregation in time, let us imagine that, for performance assurance, the network management application must receive the performance metrics associated with a specific flow, every millisecond. Since the performance metrics will be constantly changing, there is a new dimension to the Flow definition: we are not dealing anymore with a single Flow lasting a few seconds or a few minutes, but with a multitude of one millisecond sub flows for which the performance metrics are reported.

Which current protocol is suitable for these requirements: push mechanism, highly granular information, and huge number of similar records? IPFIX, as specified in RFC5101 would give part of the solution.

2.1. The IPFIX Track

The IPFIX working group has specified a protocol to export IP Flow information [RFC5101]. 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.).

The IPFIX protocol specification [RFC5101] specifies that IP traffic measurements for Flows are exported using a TLV (type, length, value) format. The information is exported using a Template Record that is sent once to export the {type, length} pairs that define the data format for the Information Elements in a Flow. The Data Records specify values for each Flow.

Based on the Requirements for IP Flow Information Export (IPFIX) [RFC3917], the IPFIX protocol has been optimized to export Flow related information. 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 [RFC5102], registered with IANA [IANA-IPFIX], or specified as an enterprise-specific Information Element. For each Information Element, the IPFIX information model [RFC5102] defines a numeric identifier, an abstract data type, an encoding mechanism for the data type, and any semantic constraints. Only basic, single-valued data types, e.g., numbers, strings, and network addresses are currently supported.

2.2. The IPFIX Limitations

The IPFIX protocol specification [RFC5101] does not support the encoding of hierarchical structured data and arbitrary-length lists (sequences) of Information Elements as fields within a Template Record. As it is currently specified, a Data Record is a "flat" list of single-valued attributes. However, it is a common data modeling requirement to compose complex hierarchies of data types, with multiple occurrences, e.g., 0..* cardinality allowed for instances of each Information Element in the hierarchy.

A typical example is the MPLS label stack entries model. An early NetFlow implementation used two Information Elements to represent the MPLS label stack entry: a "label stack entry position" followed by a "label stack value". However, several drawbacks were discovered. Firstly, the Information Elements in the Template Record had to be imposed so that the position would always precede the value. However, some encoding optimizations are based on the permutation of Information Element order. Secondly, a new semantic intelligence, not described in the information model, had to be hardcoded in the Collecting Process: the label value at the position "X" in the stack is contained in the "label stack value" Information Element following by a "label stack entry position" Information Element containing the value "X". Therefore, this model was abandoned.

The selected solution in the IPFIX information model [RFC5102] is a long series of Information Elements: mplsTopLabelStackSection, mplsLabelStackSection2, mplsLabelStackSection3, mplsLabelStackSection4, mplsLabelStackSection5, mplsLabelStackSection6, mplsLabelStackSection7, mplsLabelStackSection8, mplsLabelStackSection9, mplsLabelStackSection10. While this model removes any ambiguity, it overloads the IPFIX information model with repetitive information. Furthermore, if mplsLabelStackSection11 is required, IANA [IANA-IPFIX] will not be able to assign the new Information Element next to the other ones in the registry, which might cause some confusion.

2.3. Structured Data Use Cases

Clearly the MPLS label stack entries issue can best be solved by using a real structured data type composed of ("label stack entry position", "label stack value") pairs, potentially repeated multiple times in Flow Records, since this would be the most efficient from an information model point of view.

Some more examples enter the same category: how to encode the list of output interfaces in a multicast Flow, how to encode the list of BGP Autonomous Systems (AS) in a BGP Flow, how to encode the BGP communities in a BGP Flow, etc?

The one-way delay passive measurement, which is described in the IPFIX Applicability [RFC5472], is yet another example that would benefit from a structured data encoding. Assuming synchronized clocks, the Collector can deduce the one-way delay between two Observation Points from the following two Information Elements, collected from two different Observation Points:

- Packet arrival time: observationTimeMicroseconds [RFC5477]
- Packet ID: digestHashValue [RFC5477]

In practice, this implies that many pairs of (observationTimeMicroseconds, digestHashValue) must be exported for each Observation Point, even if Hash-Based Filtering [RFC5475] is used. On top of that information, if the requirement is to understand the one-way delay per application type, the 5-tuple (source IP address, destination IP address, protocol, source port, destination port) would need to be added to every Flow Record. Instead of exporting this repetitive 5-tuple, as part of every single Flow Record a Flow Record composed of a structured data type such as the following would save a lot of bandwidth:

```
5-tuple
          { observationTimeMicroseconds 1, digestHashValue 1 }
          { observationTimeMicroseconds 2, digestHashValue 2 }
          { observationTimeMicroseconds 3, digestHashValue 3 }
          { ...  , ... }
```

As a last example, here is a more complex case of hierarchical structured data encoding. Consider the example scenario of an IPS (Intrusion Prevention System) alert data structure containing multiple participants, where each participant contains multiple attackers and multiple targets, with each target potentially composed of multiple applications, as depicted below:

```
alert
    signatureId
    protocolIdentifier
    riskRating
    participant 1
        attacker 1
            sourceIPv4Address
            applicationId
```

```
             ...
             attacker N
                 sourceIPv4Address
                 applicationId
             target 1
                 destinationIPv4Address
                 applicationId 1
                 ...
                 applicationId n
             ...
             target N
                 destinationIPv4Address
                 applicationId 1
                 ...
                 applicationId n
         participant 2
             ...
```

To export this information in IPFIX, the data would need to be flattened (thus losing the hierarchical relationships) and a new IPFIX Template created for each alert, according to the number of applicationId elements in each target, the number of targets and attackers in each participant, and the number of participants in each alert. Clearly each Template will be unique to each alert, and a large amount of CPU, memory and export bandwidth will be wasted creating, exporting, maintaining, and withdrawing the Templates. See Appendix C for a specific example related to this case study.

2.4. The Proposal

This document specifies an IPFIX extension to support hierarchical structured data and variable-length lists by defining three new Information Elements and three corresponding new abstract data types called basicList, subTemplateList, and subTemplateMultiList. These are defined in Section 4.1.

The three Structured Data Information Elements carry some semantic information so that the Collecting Process can understand the relationship between the different list elements. The semantic in the Structured Data Information Elements is provided in order to express the relationship among the multiple top-level list elements. As an example, if a list is composed of the elements

(A,B,C), the semantic expresses the relationship among A, B, and C, regardless of whether A, B, and C, are individual elements or list of elements.

It is important to note that whereas the Information Elements and abstract data types defined in the IPFIX information model [RFC5102] represent single values, these new abstract data types are structural in nature and primarily contain references to other Information Elements and to Templates. By referencing other Information Elements and Templates from an Information Element's data content, it is possible to define complex data structures such as variable-length lists and to specify hierarchical containment relationships between Templates. Therefore, this document prefers the more generic "Data Record" term to the "Flow Record" term.

This document specifies three new abstract data types, which are basic blocks to represent structured data. However, this document does not comment on all possible combinations of basicList, subTemplateList, and subTemplateMultiList. Neither, does it limit the possible combinations.

3. Terminology

IPFIX-specific terminology used in this document is defined in Section 2 of the IPFIX protocol specification [RFC5101] and Section 3 of PSAMP protocol specification [RFC5476]. As in [RFC5101], these IPFIX-specific terms have the first letter of a word capitalized when used in this document.

3.1. New Terminology

Structured Data Information Element

One of the Information Elements supporting structured data, i.e., the basicList, subTemplateList, or subTemplateMultiList Information Elements specified in section 4.3.

4. Linkage with the IPFIX Information Model

As in the IPFIX Protocol specification [RFC5101], the new Information Elements specified in Section 4.3. below MUST be sent in canonical format in network-byte order (also known as the big-endian byte ordering).

4.1. New Abstract Data Types

This document specifies three new abstract data types, as described below.

4.1.1. basicList

The type "basicList" represents a list of zero or more instances of any Information Element, primarily used for single-valued data types. For example, a list of port numbers, a list of interface indexes, a list of AS in a BGP AS-PATH, etc.

4.1.2. subTemplateList

The type "subTemplateList" represents a list of zero or more instances of a structured data type, where the data type of each list element is the same and corresponds with a single Template Record. For example, a structured data type composed of multiple pairs of ("MPLS label stack entry position", "MPLS label stack value"), a structured data type composed of performance metrics, a structured data type composed of multiple pairs of IP address, etc.

4.1.3. subTemplateMultiList

The type "subTemplateMultiList" represents a list of zero or more instances of a structured data type, where the data type of each list element can be different and corresponds with different template definitions. For example, a structured data type composed of multiple access-list entries, where entries can be composed of different criteria types.

4.2. New Data Type Semantic

This document specifies a new data type semantic, in addition to the ones specified in the section 3.2 of the IPFIX information model [RFC5102], as described below.

4.2.1. List

A list represents an arbitrary-length sequence of zero or more structured data Information Elements, either composed of regular Information Elements or composed of data conforming to a Template Record.

4.3. New Information Elements

This document specifies three new Information Elements, as described below.

4.3.1. basicList

A basicList specifies a generic Information Element with a basicList abstract data type as defined in Section 4.1.1. and list semantics as defined in Section 4.2.1. For example, a list of port numbers, a list of interface indexes, etc.

EDITOR'S NOTE: while waiting for IANA [IANA-IPFIX] to assign this new Information Element identifier, the value XXX is used in all the examples and in the XML in Appendix A.

4.3.2. subTemplateList

A subTemplateList specifies a generic Information Element with a subTemplateList abstract data type as defined in Section 4.1.2. and list semantics as defined in Section 4.2.1.

EDITOR'S NOTE: while waiting for IANA [IANA-IPFIX] to assign this new Information Element identifier, the value YYY is used in all the examples and in the XML in Appendix A.

4.3.3. subTemplateMultiList

A subTemplateMultiList specifies a generic Information Element with a subTemplateMultiList abstract data type as defined in Section 4.1.3. and list semantics as defined in Section 4.2.1.

EDITOR'S NOTE: while waiting for IANA [IANA-IPFIX] to assign this new Information Element identifier, the value ZZZ is used in all the examples and in the XML in Appendix A.

4.4. New Structured Data Type Semantics

Structured data type semantics are provided in order to express the relationship among multiple list elements in a Structured Data Information Element. These structured data type semantics require a new IPFIX subregistry, as specified in the "IANA Considerations" section. The semantics are specified in the next following subsections.

4.4.1. undefined

The "undefined" structured data type semantic specifies that the semantic of list elements is not specified, and that, if a semantic exists, then it is up to the Collecting Process to draw its own conclusions. The "undefined" structured data type semantic is the default structured data type semantic.

4.4.2. noneOf

The "noneOf" structured data type semantic specifies that none of the elements are actual properties of the Data Record.

For example, a mediator might want to report to a Collector that a specific Flow is suspicious, but that it checked already that this Flow does not belong to the attack type 1, attack type 2, and attack type 3. So this Flow might need some further inspection. In such a case, the mediator would report the Flow Record with a basicList composed of (attack type 1, attack type 2, attack type 3) and the respective structured data type semantic of "noneOf".

Another example is a router that monitors some specific BGP AS-PATHs and reports if a Flow belongs to any of them. If the router wants to export that a Flow does not belong to any of the monitored BGP AS-PATHs, the router reports a Data Record with a basicList composed of (BGP AS-PATH 1, BGP AS-PATH 2, BGP AS-PATH 3) and the respective structured data type semantic of "noneOf".

4.4.3. exactlyOneOf

The "exactlyOneOf" structured data type semantic specifies that only a single element from the structured data is an actual property of the Data Record. This is equivalent to a logical XOR operation.

For example, if a Flow record contains a basicList of outgoing interfaces with the "exactlyOneOf" semantic, then it implies that the reported Flow only egressed from a single interface, although the Flow Record lists all of the possible outgoing interfaces. This is a typical example of a per destination load-balancing.

Another example is a mediator that must aggregate Data Records from different Observation Points and report an aggregated Observation Point. However, the different Observation Points can be specified by different Information Element types depending on the Exporter. For example:

Exporter1 Observation Point is characterized by the exporterIPv4Address, so a specific Exporter can be represented.

Exporter2 Observation Point is characterized by the exporterIPv4Address and a basicList of ingressInterface, so the Exporting Process can express that the observations were made on a series of input interfaces.

Exporter3 Observation Point is characterized by the exporterIPv4Address and a specific lineCardId, so the Exporting Process can express that the observation was made on a specific line card.

If the mediator models the three different types of Observation Points with the three Template Records below:

```
    Template Record 1: exporterIPv4Address
    Template Record 2: exporterIPv4Address, basicList of
                       ingressInterface
    Template Record 3: exporterIPv4Address, lineCardId
```

then it can represent the aggregated Observation Point with a subTemplateMultiList and the semantic "exactlyOneOf". The aggregated 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. This implies that the Flow was observed at exactly one of the Observation Points reported.

4.4.4. oneOrMoreOf

The "oneOrMoreOf" structured data type semantic specifies that one or more elements from the list in the structured data are actual properties of the Data Record. This is equivalent to a logical OR operation.

Consider an example where a mediator must report an aggregated Flow (e.g. by aggregating IP addresses from IP prefixes), with an aggregated Observation Point. However, the different Observation Points can be specified by different Information Element types as described in Section 4.4.2.

If the mediator models the three different types of Observation Points with the three Template Records below:

```
Template Record 1: exporterIPv4Address
Template Record 2: exporterIPv4Address, basicList of
                   ingressInterface
Template Record 3: exporterIPv4Address, lineCardId
```

then it can represent the aggregated Observation Point with a subTemplateMultiList and the semantic "oneOrMoreOf". The aggregated Observation Point is modeled with the Data Records corresponding to either Template Record 1, Template Record 2, or Template Record 3. This implies that the Flow was observed on at least one of the Observation Points reported, and potentially on multiple Observation Points.

4.4.5. allOf

The "allOf" structured data type semantic specifies that all of the list elements from the structured data are actual properties of the Data Record.

For example, if a Record contains a basicList of outgoing interfaces with the "allOf" semantic, then the observed Flow is typically a multicast Flow where each packet in the Flow has been replicated to each outgoing interface in the basicList.

4.4.6. ordered

The "ordered" structured data type semantic specifies that elements from the list in the structured data are ordered.

For example, an Exporter might want to export the AS10 AS20 AS30 AS40 BGP AS-PATH. In such a case, the Exporter would report a basicList composed of (AS10, AS20, AS30, AS40) and the respective structured data type semantic of "ordered".

4.5. Encoding of IPFIX Data Types

The following subsections define the encoding of the abstract data types defined in Section 4.1. above. These data types may be encoded using either fixed or variable-length Information Elements, as discussed in Section 5.1. .

4.5.1. basicList

The basicList Information Element defined in Section 4.3.1. represents a list of zero or more instances of an Information Element and is encoded 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Semantic    |0|          Field ID           |   Element...  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| ...Length     |           basicList Content ...               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure A: basicList Encoding

Semantic

The Semantic field indicates the relationship among the different Information Element values within this Structured Data Information Element.

Field ID

Field ID is the Information Element identifier of the Information Element(s) contained in the list.

Element Length

The Element Length field indicates the length of each list element specified by Field ID, or contains the value 0xFFFF if the length is encoded as a variable-length Information Element at the start of the basicList Content, per Section 7 of [RFC5101].

The Element Length field is effectively part of the header, so even in the case of a zero-element list, it MUST NOT be omitted.

basicList Content

A Collecting Process decodes list elements from the basicList Content until no further data remains. A field count is not included but can be derived when the Information Element is decoded.

Note that in the diagram above, Field ID is shown with the Enterprise bit (most significant bit) set to 0. If instead the Enterprise bit is set to 1, a four-byte Enterprise Number MUST be encoded immediately after the Element Length as shown below. See the "Field Specifier Format" section in the IPFIX Protocol [RFC5101] for additional information.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Semantic   |1|         Field ID            |   Element...  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| ...Length     |               Enterprise Number ...           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      ...      |              basicList Content ...            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure B: basicList Encoding with Enterprise Number

Also note that, if a basicList has zero elements, the encoded data contains the Semantic field, Field ID, the Element Length field and the four-byte Enterprise Number (if present), while basicList Content is empty.

If the basicList is encoded as a variable-length Information Element in less than 255 octets, it is encoded with the Length field per Section 7 of [RFC5101] 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Length (< 255)|          basicList Encoding ...               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      ... continuing as needed                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure C: Variable-Length basicList Encoding (Length < 255 octets)

If the basicList is encoded as a variable-length Information Element in 255 or more octets, it is encoded with the Length field per Section 7 of [RFC5101] 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      255      |      Length (0 to 65535)      |       ...     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     ... basicList Encoding                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure D: Variable-Length basicList Encoding (Length 0 to 65535 octets)

4.5.2. subTemplateList

The subTemplateList Information Element represents a list of zero or more Data Records corresponding to a specific Template. Because the Template Record referenced by a subTemplateList Information Element can itself contain other subTemplateList Information Elements, and because these Template Record references are part of the Information Elements content in the Data Record, it is possible to represent complex hierarchical data structures. The following diagram shows how a subTemplateList Information Element is encoded within a Data Record:

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Semantic    |         Template ID           |      ...      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                subTemplateList Content    ...                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
 |                              ...                              |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure E: subTemplateList Encoding

Semantic

The Semantic field indicates the relationship among the different Data Records within this Structured Data Information Element.

Template ID

The Template ID field contains the ID of the template used to encode and decode the subTemplateList Content.

subTemplateList Content

subTemplateList Content consists of zero or more instances of Data Records corresponding to the Template ID specified in the Template ID Field. A Collecting Process decodes the subTemplateList Content until no further data remains. A record count is not included but can be derived when the subTemplateList is decoded. Encoding and decoding are performed recursively if the specified Template itself contains Structured Data Information Elements as described here.

Note that, if a subTemplateList has zero elements, the encoded data contains only the Semantic field and the Template ID field, while subTemplateList Content is empty.

If the subTemplateList is encoded as a variable-length Information Element in less than 255 octets, it is encoded with the Length field per Section 7 of [RFC5101] 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
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | Length (< 255)|          subTemplateList Encoding ...         |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                    ... continuing as needed                   |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure F: Variable-Length subTemplateList Encoding (Length < 255 octets)

If the subTemplateList is encoded as a variable-length Information Element in 255 or more octets, it is encoded with the Length field per Section 7 of [RFC5101] 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      255      |      Length (0 to 65535)      |      ...      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    ... subTemplateList Encoding               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure G: Variable-Length subTemplateList Encoding (Length 0 to 65535 octets)

4.5.3. subTemplateMultiList

Whereas each element in a subTemplateList Information Element corresponds to a single Template, it is sometimes useful for a list to contain elements corresponding to different Templates. To support this case, each top-level element in a subTemplateMultiList Information Element carries a Template ID, Length and zero or more Data Records corresponding to the Template ID. The following diagram shows how a subTemplateMultiList Information Element is encoded within a Data Record. Note that the encoding following the Semantic field is consistent with the Set Header specified in [RFC5101].

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Semantic   |         Template ID X         |Data Records...|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| ... Length X  |        Data Record X.1 Content ...            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               ...                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      ...      |        Data Record X.2 Content ...            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               ...                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      ...      |        Data Record X.L Content ...            |
```

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      ...      |         Template ID Y         |Data Records...|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| ... Length Y  |        Data Record  Y.1 Content ...           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      ...      |        Data Record Y.2 Content ...            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      ...      |        Data Record Y.M Content ...            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      ...      |         Template ID Z         |Data Records...|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| ... Length Z  |        Data Record Z.1 Content ...            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      ...      |        Data Record Z.2 Content ...            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      ...      |        Data Record Z.N Content ...            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      ...      |
+-+-+-+-+-+-+-+-+
```

Figure H: subTemplateMultiList Encoding

Semantic

The Semantic field indicates the top-level relationship among the series of Data Records corresponding to the different Template Records within this Structured Data Information Element.

Template ID

Unlike the subTemplateList Information Element, each element of the subTemplateMultiList contains a Template ID which specifies the encoding of the following Data Records.

Data Records Length

The total length of the Data Records encoding for the Template ID previously specified, including the 2 bytes for the Template ID and the 2 bytes for the Data Records Length field itself.

Data Record X.M

The Data Record X.M consists of the Mth Data Record of the Template Record X. A Collecting Process decodes the Data Records according to Template Record X until no further data remains, according to the Data Records Length X. Further Template IDs and Data Records may then be decoded according to the overall subTemplateMultiList length. A record count is not included but can be derived when the Element Content is decoded. Encoding and decoding are performed recursively if the specified Template itself contains Structured Data Information Elements as described here.

In the exceptional case of zero instances in the subTemplateMultiList, no data is encoded, only the Semantic field and Template ID field(s), and the Data Record Length field is set to zero.

If the subTemplateMultiList is encoded as a variable-length Information Element in less than 255 octets, it is encoded with the Length field per Section 7 of [RFC5101] 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Length (< 255)|         subTemplateMultiList Encoding ...     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      ... continuing as needed                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure I: Variable-Length subTemplateMultiList Encoding (Length < 255 octets)

If the subTemplateMultiList is encoded as a Variable-Length Information Element in 255 or more octets, it is encoded with the Length field per Section 7 of [RFC5101] 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      255      |      Length (0 to 65535)      |       ...     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 ... subTemplateMultiList Encoding             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure J: Variable-Length subTemplateMultiList Encoding (Length 0 to 65535 octets)

5. Structured Data Format

5.1. Length Encoding Considerations

The new Structured Data Information Elements represent a list that potentially carries complex hierarchical and repeated data.

When the encoding of a Structured Data Information Element has a fixed length (because, for example, it contains the same number of fixed-length elements, or if the permutations of elements in the list always produces the same total length), the element length can be encoded in the corresponding Template Record.

However, when representing variable-length data, hierarchical data, and repeated data with variable element counts, where the number and length of elements can vary from record to record, we RECOMMEND that the Information Elements are encoded using the variable-length encoding described in Section 7 of [RFC5101], with the length carried before the Structured Data Information Element encoding.

Because of the complex and repeated nature of the data, it is potentially difficult for the Exporting Process to efficiently know in advance the exact encoding size. In this case, the Exporting Process may encode the available data starting at a fixed offset and fill in the final length afterwards. Therefore, the three-byte length encoding is RECOMMENDED for variable-length information elements in all Template Records containing a Structured Data Information Element, even if the encoded length can be less than 255 bytes, because the starting offset of the data is known in advance.

When encoding such data, an Exporting Process MUST take care to not exceed the maximum allowed IPFIX message length of 65535 bytes as specified in [RFC5101].

5.2. Recursive Structured Data

It is possible to define recursive relationships between IPFIX structured data instances, for example when representing a tree structure. The simplest case of this might be a basicList where each element is itself a basicList, or a subTemplateList where one of the fields of the referenced template is itself a subTemplateList referencing the same Template. Also, the Exporting Process MUST take care when encoding recursively-defined structured data, not to exceed the maximum allowed length of an IPFIX Message (as noted in Length Encoding Considerations).

5.3. Structured Data Information Elements Applicability in Options Template Sets

Structured Data Information Elements MAY be used in Options Template Sets.

As an example, consider a mediation function that must aggregate Data Records from multiple Observation Point types:

```
Router 1, (interface 1)
Router 2, (line card A)
Router 3, (line card B)
Router 4, (line card C, interface 2)
```

In order to encode the PSAMP Selection Sequence Report Interpretation [RFC5476], the mediation function must express this combination of Observation Points as a single new Observation Point. Recall from [RFC5476] that the PSAMP Selection Sequence Report Interpretation consists of the following fields:

```
Scope:     selectionSequenceId
Non-Scope: one Information Element mapping the Observation Point
           selectorId (one or more)
```

Without structured data, there is clearly no way to express the complex aggregated Observation Point as "one Information Element mapping the Observation Point". However, the desired result may be easily achieved using the structured data types. Refer to Section 8.5. for an encoding example related to this case study.

Regarding the scope in the Options Template Record, the IPFIX specification [RFC5101] mentions that "The IPFIX protocol doesn’t prevent the use of any Information Elements for scope". Therefore, a Structured Data Information Element MAY be used as scope in an Options Template Set.

Extending the previous example, the mediation function could export a given name for this complex aggregated Observation Point:

```
Scope: Aggregated Observation Point (structured data)
Non-Scope: a new Information Element containing the name
```

5.4. Usage Guidelines for Equivalent Data Representations

Because basicList, subTemplateList, and subTemplateMultiList are all lists, in several cases there is more than one way to represent what is effectively the same data structure. However, in some cases, one approach has an advantage over the other e.g. more compact, uses fewer resources, etc., and is therefore preferred over an alternate representation.

A subTemplateList can represent the same simple list of single-value Information Elements as a basicList, if the Template referenced by the subTemplateList contains only one single-valued Information Element. Although the encoding is more compact than a basicList by two bytes, using a subTemplateList in this case requires a new Template per Information Element. The basicList requires no additional Template and is therefore RECOMMENDED in this case.

Although a subTemplateMultiList with one Element can represent the contents of a subTemplateList, the subTemplateMultiList carries two additional bytes (Element Length). It is also potentially useful to a Collecting Process to know in advance that a subTemplateList directly indicates that list element types are consistent. The subTemplateList Information Element is therefore RECOMMENDED in this case.

The Semantic field in a subTemplateMultiList indicates the top-level relationship among the series of Data Records corresponding to the different Template Records, within this Structured Data

Information Element. If a semantic is required to describe the relationship among the different Data Records corresponding to a single Template ID within the subTemplateMultiList, then an encoding based on a basicList of subTemplateLists should be used, refer to Section 5.6 for more information. Alternatively, if a semantic is required to describe the relationship among all Data Records within a subTemplateMultiList (regardless of the Template Record), an encoding based on a subTemplateMultiList with one Data Record corresponding to a single Template ID can be used.

Note that the referenced Information Element(s) in the Structured Data Information Elements can be taken from the IPFIX information model [RFC5102], the PSAMP information model [RFC5477], any of the Information Elements defined in the IANA IPFIX registry [IANA-IPFIX] or enterprise-specific Information Elements.

5.5. Padding

The Exporting Process MAY insert some padding octets in structured data field values in a Data Record by including the 'paddingOctets' Information Element as described in [RFC5101] Section 3.3.1. The paddingOctets Information Element can be included in a Template Record referenced by structured data Information Element for this purpose.

5.6. Semantic

Semantic interpretations of received Data Records at or beyond the Collecting Process remain explicitly undefined, unless that data is transmitted using this extension with explicit Structured Data type semantic information.

It is not the Exporter's role to check the validity of the semantic representation of Data Records. Therefore, the Exporter SHOULD NOT check all semantically meaningless combinations before exporting the Data Record.

More complex semantics can be expressed as a combination of the Semantic Data Information Elements specified in this document.

For example, the export of the AS10 AS20 AS30 AS40 {AS50,AS60} BGP AS-PATH would be reported as a basicList of two elements, each element being a basicList of BGP AS, with the top level structured data type semantic of "ordered". The first element would contain a basicList composed of (AS10,AS20,AS30,AS40) and the respective structured data type semantic of "ordered", while the second element would contain a basicList composed of (AS50, AS60) and the respective structured data type semantic of "exactlyOneOf". A high level Data Record diagram would be represented as:

```
     BGP AS-PATH = (basicList, ordered,

         (basicList, ordered, AS10,AS20,AS30,AS40),

         (basicList, exactlyOneOf, AS50, AS60)

     )
```

If a semantic is required to describe the relationship among the different Data Records corresponding to a single Template ID within the subTemplateMultiList, then an encoding based on a basicList of subTemplateLists should be used, as shown in the next case study.

Case study 1:

In this example, an Exporter monitoring security attacks must export a list of security events consisting of attackers and targets. For the sake of the example, assume that the Collector can differentiate the attacker (which is expressed using source fields) from the target (which is expressed using destination fields). Imagine that attackers A1 or A2 may attack targets T1 and T2.

The first case uses a subTemplateMultiList composed of two Template Records, one representing the attacker and one representing the target, each of them containing an IP address and a port.

```
     Attacker Template Record = (src IP address, src port)

     Target Template Record = (dst IP address, dst port)
```

A high level Data Record diagram would be represented as:

```
      Alert = (subTemplateMultiList, allOf,

         (Attacker Template Record, A1, A2),

         (Target Template Record, T1, T2)

      )
```

The Collecting Process can only conclude that the list of attackers (A1, A2) and the list of targets (T1, T2) are present, without knowing the relationship amongst attackers and targets. The Exporting Process would have to explicitly call out the relationship amongst attackers and targets as the top level semantic offered by the subTemplateMultiList isn't sufficient.

The only proper encoding for the previous semantic (i.e. attacker A1 or A2 may attack target T1 and T2) uses a basicList of subTemplateLists and is represented as follows:

```
     Attacker Template Record = (src IP address, src port)

     Target Template Record = (dst IP address, dst port)

     Alert = (basicList, allof,

           (subTemplateList, exactlyOneOf, attacker A1, A2)

           (subTemplateList, allOf, target T1, T2)

     )
```

Case study 2:

In this example, an Exporter monitoring security attacks must export a list of attackers and targets. For the sake of the example, assume that the Collector can differentiate the attacker (which is expressed using source fields) from the target (which is expressed using destination fields). Imagine that attackers A1 or A2 are attacking target T1, while attacker A3 is attacking targets T2 and T3. The first case uses a subTemplateMultiList that contains Data Records corresponding to two Template Records, one representing the attacker and one representing the target, each of them containing an IP address and a port.

```
     Attacker Template Record = (src IP address, src port)

     Target Template Record = (dst IP address, dst port)
```

A high level Data Record diagram would be represented as:

```
      Alert = (subTemplateMultiList, allOf,

         (Attacker Template Record, A1, A2, A3),
```

```
         (Target Template Record, T1, T2, T3)

      )
```

The Collecting Process can only conclude that the list of attackers (A1, A2, A3) and the list of targets (T1, T2, T3) are present, without knowing the relationship amongst attackers and targets.

The second case could use a Data Record definition composed of the following:

```
      Alert = (subTemplateMultiList, allOf,

         (Attacker Template Record, A1, A2),

         (Target Template Record, T1),

         (Attacker Template Record, A3),

         (Target Template Record, T2, T3)

      )
```

With the above representation, the Collecting Process can infer that the alert consists of the list of attackers (A1, A2), target (T1), attacker (A3) and list of targets (T2, T3).   From the sequence in which attackers and targets are encoded, the Collector can possibly deduce that some relationship exists among (A1, A2, T1) and (A2, T1, T2) but cannot understand what it is exactly. So, there is a need for the Exporting Process to explicitly define the relationship between the attackers and targets and the top-level semantic of the subTemplateMultiList is not sufficient.

The only proper encoding for the previous semantic (i.e. attacker A1 or A2 attack target T1, attacker A3 attacks targets T2 and T3) uses a basicList of subTemplateLists and is represented as follows:

```
     Participant P1 =

     (basicList, allOf,

           (subTemplateList, exactlyOneOf, attacker A1, A2)

           (subTemplateList, undefined, target T1)
```

```
     )

     Participant P2 =

     (basicList, allOf,

           (subTemplateList, undefined, attacker A3,

           (subTemplateList, allOf, targets T2, T3)

     )
```

The security alert is represented as a subTemplateList of participants.

```
     Alert =

        (subTemplateList, allOf, Participant P1, Participant P2)
```

Note that, in the particular case of a single element in a Structured Data Information Element, the semantic field is actually not very useful since it specifies the relationship among multiple elements. Any choice of allOf, exactlyOneOr, or OneOrMoreOf would provide the same result semantically. Therefore, in case of a single element in a Structured Data Information Element, the default "undefined" semantic SHOULD be used.

6. Template Management

This section introduces some more specific Template management and Template Withdrawal Message-related specifications compared to the IPFIX protocol specification [RFC5101].

First of all, the Template ID uniqueness is unchanged compared to [RFC5101]; the uniqueness is local to the Transport Session and Observation Domain that generated the Template ID. In other words, the Set ID used to export the Template Record does not influence the Template ID uniqueness.

While [RFC5101] mentions that: "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.", this rule MAY be ignored within Structured Data Information Elements.

As specified in [RFC5101], Templates that are not used anymore SHOULD be deleted. Deleting a Template implies that it MUST NOT be used within subTemplateList and subTemplateMultiList any more. Before reusing a Template ID, the Template MUST be deleted. In order to delete an allocated Template, the Template is withdrawn through the use of a Template Withdrawal Message.

7. The Collecting Process’s Side

This section introduces some more specific specifications to the Collection Process compared to Section 9 in the IPFIX Protocol [RFC5101].

As described in [RFC5101], a 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. Therefore a Collection Process that does not support the extension specified in this document can ignore the Structured Data Information Elements in a Data Record, or it can ignore Data Records containing these new Structured Data Information Elements while continuing to process other Data Records.

If the structured data contains the "undefined" structured data type semantic, the Collecting Process MAY attempt to draw its own conclusion in terms of the semantic contained in the Data Record.

8. Structured Data Encoding Examples

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

8.1. Encoding a Multicast Data Record with basicList

Consider encoding a multicast Data Record containing the following data:

Ingress If	Source IP	Destination IP	Egress Interfaces
9	192.0.2.201	233.252.0.1	1, 4, 8

Template Record for the multicast Flows, with the Template ID 256:

```
  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 = 2            |      Length = 24 octets       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |       Template ID = 256       |       Field Count = 4         |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0|    ingressInterface = 10    |       Field Length = 4        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0|   sourceIPv4Address = 8     |       Field Length = 4        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0| DestinationIPv4Address = 12 |       Field Length = 4        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0|       basicList = XXX       |     Field Length = 0xFFFF     |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure K: Encoding basicList, Template Record

The list of outgoing interfaces is represented as a basicList with semantic allOf, and the Length of the list is chosen to be encoded in three bytes even though it may be less than 255 octets.

The Data Set is represented 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 = 256         |          Length = 36          |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                      ingressInterface = 9                     |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                sourceIPv4Address = 192.0.2.201                |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |              DestinationIPv4Address = 233.252.0.1             |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |      255      |         List Length = 17      | semantic=allOf|
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | egressInterface FieldId = 14  |egressInterface Field Length=4 |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                egressInterface value 1 = 1                    |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                egressInterface value 2 = 4                    |
```

```
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                egressInterface value 3 = 8                    |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure L: Encoding basicList, Data Record, Semantic allOf

In the example above, the basicList contains fixed-length elements. To illustrate how variable-length elements would be encoded, the same example is shown below with variable-length interface names in the basicList instead:

```
  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 = 256         |          Length = 44          |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                     ingressInterface = 9                      |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |               sourceIPv4Address = 192.0.2.201                 |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |             DestinationIPv4Address = 233.252.0.1              |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |      255      |        List Length = 25       | semantic=allOf|
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0| InterfaceName FieldId = 82  | InterfaceName Field Len=0xFFFF|
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |  Length = 5   |      'F'      |      'E'      |      '0'      |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |      '/'      |      '0'      |  Length = 7   |      'F'      |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |      'E'      |      '1'      |      '0'      |      '/'      |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |      '1'      |      '0'      |  Length = 5   |      'F'      |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |      'E'      |      '2'      |      '/'      |      '2'      |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure M: Encoding basicList, Data Record with Variable-Length Elements, Semantic allOf

8.2. Encoding a Load-balanced Data Record with a basicList

Consider encoding a load-balanced Data Record containing the following data:

Ingress If	Source IP	Destination IP	Egress Interfaces
9	192.0.2.201	233.252.0.1	1, 4, 8

So the Data Record egressed from either interface 1, 4, or 8. The Data Set is represented 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 = 256         |          Length = 36          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     ingressInterface = 9                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               sourceIPv4Address = 192.0.2.201                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             DestinationIPv4Address = 233.252.0.1              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      255      |        List Length = 17       |sem=exactlyOne |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| egressInterface FieldId = 14  |egressInterface Field Length=4 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                egressInterface value 1 = 1                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                egressInterface value 2 = 4                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                egressInterface value 3 = 8                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure N: Encoding basicList, Data Record, Semantic ExactlyOneOf

8.3. Encoding subTemplateList

As explained in Section 2.2. , multiple pairs of (observationTimeMicroseconds, digestHashValue) must be collected from two different Observation Points to passively compute the one-way delay across the network. This data can be exported with an optimized Data Record that consists of the following attributes:

```
    5-tuple
               { observationTimeMicroseconds 1, digestHashValue 1 }
               { observationTimeMicroseconds 2, digestHashValue 2 }
```

```
               { observationTimeMicroseconds 3, digestHashValue 3 }
               { ...  , ... }
```

A subTemplateList is best suited for exporting the list of (observationTimeMicroseconds, digestHashValue). For illustration purposes, the number of elements in the list is 5; in practice, it could be more.

```
-------------------------------------------------------------------
srcIP     | dstIP       | src   | dst  |proto| one-way delay
          |             | Port  | Port |     |   metrics
-------------------------------------------------------------------
192.0.2.1  192.0.2.105   1025     80     6    Time1, 0x0x91230613
                                              Time2, 0x0x91230650
                                              Time3, 0x0x91230725
                                              Time4, 0x0x91230844
                                              Time5, 0x0x91230978
-------------------------------------------------------------------
```

The following Template is defined for exporting the one-way delay metrics:

```
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 = 2             |      Length = 16 octets       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Template ID = 257       |       Field Count = 2         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0| observationTimeMicroSec=324 |       Field Length = 8        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|   digestHashValue = 326     |       Field Length = 4        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure O: Encoding subTemplateList, Template for One-Way Delay Metrics

The Template Record for the Optimized Data Record 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 = 2            |      Length = 32 octets       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Template ID = 258       |       Field Count = 6         |
```

```
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0|   sourceIPv4Address = 8     |       Field Length = 4        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0| destinationIPv4Address = 12 |       Field Length = 4        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0|  sourceTransportPort = 7    |       Field Length = 2        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0| destinationTransportPort= 11|       Field Length = 2        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0| protocolIdentifier = 4      |       Field Length = 1        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0|  subTemplateList = YYY      |     Field Length = 0xFFFF     |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure P: Encoding subTemplateList, Template Record

The list of (observationTimeMicroseconds, digestHashValue) is exported as a subTemplateList with semantic allOf. The Length of the subTemplatelist is chosen to be encoded in three bytes even though it may be less than 255 octets.

The Data Record is represented 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 = 258          |      Length = 83 octets       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                sourceIPv4Address = 192.0.2.1                  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |              destinationIPV4Address = 192.0.2.105             |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | sourceTransportPort = 1025    | destinationTransportPort = 80 |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | Protocol = 6  |      255      | one-way metrics list len = 63 |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | semantic=allOf|       TemplateID = 257        | TimeValue1    |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                 ... octets 2-5 of TimeValue1                  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |          ... octets 6-8 of TimeValue1        |digestHashVal1=|
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                ... 0x0x91230613               | TimeValue2    |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                 ... octets 2-5 of TimeValue2                  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |          ... octets 6-8 of TimeValue2        |digestHashVal2=|
```

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                ... 0x0x91230650             | TimeValue3    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 ... octets 2-5 of TimeValue3                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          ... octets 6-8 of TimeValue3       |digestHashVal3=|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                ... 0x0x91230725             | TimeValue4    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 ... octets 2-5 of TimeValue4                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          ... octets 6-8 of TimeValue4       |digestHashVal4=|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                ... 0x0x91230844             | TimeValue5    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 ... octets 2-5 of TimeValue5                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          ... octets 6-8 of TimeValue5       |digestHashVal5=|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                ... 0x0x91230978             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure Q: Encoding subTemplateList, Data Set

8.4. Encoding subTemplateMultiList

As explained in Section 4.5.3., a subTemplateMultiList is used to export a list of mixed-type content where each top level element corresponds to a different Template Record.

To illustrate this, consider the Data Record with the following attributes:

```
5-tuple (Flow Keys), octetCount, packetCount
          attributes for filtering
               selectorId,
               selectorAlgorithm
          attributes for sampling
               selectorId,
               selectorAlgorithm,
               samplingPacketInterval,
               samplingPacketSpace
```

This example demonstrates that the Selector Report Interpretation [RFC5476] can be encoded with the subTemplateMultiList. More specifically, the example describes Property Match Filtering Selector Report Interpretation [RFC5476] used for filtering purposes, and the Systemic Count-Based Sampling as described in Section 6.5.2.1 of [RFC5476]. Some traffic will be filtered according to match properties configured, some will be sampled, some will be filtered and sampled, and some will not be filtered or be sampled.

A subTemplateMultiList is best suited for exporting this variable data. A Template is defined for filtering attributes and another Template is defined for sampling attributes. A Data Record can contain data corresponding to either of the Templates, both of them, or neither of them.

Consider the example below where the following Data Record contains both filtering and sampling attributes.

Key attributes of the Data Record:

srcIP	dstIP	src Port	dst Port	proto	octetCount	packet Count
192.0.2.1	192.0.2.105	1025	80	6	108000	120

Filtering attributes:

selectorId	selectorAlgorithm
100	5 (Property Match Filtering)

Sampling attributes:

For Systemic Count-Based Sampling as defined in Section 6.5.2.1 of [RFC5476] the required algorithm-specific Information Elements are:

```
samplingPacketInterval: number of packets selected in a row
samplingPacketSpace:    number of packets between selections
```

Example of a simple 1 out-of 100 systematic count-based Selector definition, where the samplingPacketInterval is 1 and the samplingPacketSpace is 99.

selectorId	selectorAlgorithm	sampling Packet Interval	sampling Packet Space
15	1 (Count-Based Sampling)	1	99

To represent the Data Record, the following Template Records are defined:

```
    Template for filtering attributes: 259
     Template for sampling attributes: 260
     Template for Flow Record: 261

     Flow record (261)
          |  (sourceIPv4Address)
          |  (destinationIPv4Address)
          |  (sourceTransportPort)
          |  (destinationTransportPort)
          |  (protocolIdentifier)
          |  (octetTotalCount)
          |  (packetTotalCount)
          |
          +------ filtering attributes (259)
          |          (selectorId)
          |          (selectorAlgorithm)
          |
          +------ sampling attributes (260)
          |          (selectorId)
          |          (selectorAlgorithm)
          |          (samplingPacketInterval)
          |          (samplingPacketSpace)
```

The following Template Record is defined for filtering attributes:

```
  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 = 2           |          Length = 16          |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |      Template ID = 259        |        Field Count = 2        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0|    selectorId = 302         |        Field Length = 4       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0| selectorAlgorithm = 304     |        Field Length = 1       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure R: Encoding subTemplateMultiList, Template for Filtering Attributes

The Template for sampling attributes is defined 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 = 2           |          Length = 24          |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |      Template ID = 260        |        Field Count = 4        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0|    selectorId = 302         |        Field Length = 4       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0|  selectorAlgorithm = 304    |        Field Length = 1       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0| samplingPacketInteval = 305 |        Field Length = 1       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0| samplingPacketSpace = 306   |        Field Length = 1       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure S: Encoding subTemplateMultiList, Template for Sampling Attributes

Note that while selectorAlgorithm is defined as unsigned16, and samplingPacketInterval and samplingPacketSpace are defined as unsigned32, they are compressed down to 1 octet here as allowed by Reduced Size Encoding in Section 6.2 of the IPFIX protocol specifications [RFC5101].

Template for the Flow Record is defined as shown below:

```
  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 = 2           |          Length = 40          |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |      Template ID = 261        |        Field Count = 8        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0|   sourceIPv4Address = 8     |       Field Length = 4        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0| destinationIPv4Address = 12 |       Field Length = 4        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0| sourceTransportPort = 7     |       Field Length = 2        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0| destinationTransportPort=11 |       Field Length = 2        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0| protocolIdentifier = 4      |       Field Length = 1        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0|   octetTotalCount = 85      |       Field Length = 4        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0|   packetTotalCount = 86     |       Field Length = 4        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0| subTemplateMultiList = ZZZ  |     Field Length = 0XFFFF     |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure T: Encoding subTemplateMultiList, Template for Flow Record

A subTemplateMultiList with semantic allOf is used to export the filtering and sampling attributes. The Length field of the subTemplateMultilist is chosen to be encoded in three bytes even though it may be less than 255 octets.

The Data Record is encoded 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 = 261            |          Length = 49          |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                 sourceIPv4Address = 192.0.2.1                 |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |              destinationIPv4Address = 192.0.2.105             |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |  sourceTransportPort = 1025   | destinationTransportPort = 80 |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | protocol = 6  |        octetTotalCount = 108000               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |     ...       |        packetTotalCount = 120                 |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |     ...       |      255      | Attributes List Length = 21   |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
 |semantic=allOf | Filtering Template ID = 259   |Filtering Attr |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | ...Length = 9 |              selectorId = ...                 |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | ...  100      |selectorAlg = 5|  Sampling Template ID = 260   |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | Sampling Attributes Length=11 |         selectorId = ...      |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |  ...         15               |selectorAlg = 1|  Interval = 1 |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | Space = 99    |
 +-+-+-+-+-+-+-+-+
```

Figure U: Encoding subTemplateMultiList, Data Set

8.5. Encoding an Options Template Set using Structured Data

As described in Section 5.3. , consider a mediation function that must aggregate Data Records from different Observation Points.

Say Observation Point 1 consists of one or more interfaces, Observation Points 2 and 3 consist of one or more line cards, and Observation Point 4 consists of one or more interfaces and one or more line cards. Without structured data, a template would have to be defined for every possible combination to interpret the data corresponding to each of the Observation Points. However, with structured data, a basicList can be used to encode the list of interfaces and another basicList can be used to encode the list of line cards.

For the sake of simplicity, each Observation Point shown below has the IP address corresponding to the Router and an <interface> or <linecard> or <line card and interface>. This can very well be extended to include a list of interfaces and a list of linecards using basicLists as explained above.

    Observation Point 1: Router 1, (interface 1)
    Observation Point 2: Router 2, (line card A)
    Observation Point 3: Router 3, (line card B)
    Observation Point 4: Router 4, (line card C, interface 2)

The mediation function wishes to express this as a single Observation Point, in order to encode the PSAMP Selection Sequence Report Interpretation (SSRI). Recall from [RFC5476] that the PSAMP Selection Sequence Report Interpretation consists of the following fields:

```
   Scope:     selectionSequenceId
   Non-Scope: one Information Element mapping the
              Observation Point
              selectorId (one or more)
```

For example, the Observation Point detailed above may be encoded in a PSAMP Selection Sequence Report Interpretation as shown below:

```
 Selection Sequence 7 (Filter->Sampling):
  Observation Point: subTemplateMultiList.
   Router 1 (IP address = 192.0.2.11), (interface 1)
   Router 2 (IP address = 192.0.2.12), (line card A)
   Router 3 (IP address = 192.0.2.13), (line card B)
   Router 4 (IP address = 192.0.2.14), (line card C, interface 2)
   selectorId: 5 (Filter, match IPV4SourceAddress 192.0.2.1)
   selectorId: 10 (Sampler, Random 1 out-of ten)
```

The following Templates are defined to represent the PSAMP SSRI:
Template for representing PSAMP SSRI: 262
Template for representing interface: 263
Template for representing linecard: 264
Template for representing linecard and interface: 265

```
    PSAMP SSRI (262)
         | (SelectionSequenceId)
         |
         +--- Observation Point 1 (263)
         |      (exporterIPv4Address)
         |      (Interface Id)
         |
         +--- Observation Point 2 and 3 (264)
         |      (exporterIPv4Address)
         |      (line card)
         |
         +--- Observation Point 4 (265)
         |      (exporterIPv4Address)
         |      (line card)
         |      (Interface Id)
         |
         | (selectorId 1)
```

```
         | (selectorId 2)
```

Note that the example could further be improved with a basicList of selectorId if many Selector IDs have to be reported.

Figure V: PSAMP SSRI to be encoded

```
   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 = 26          |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |        Template ID = 262      |         Field Count = 4       |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |     Scope Field Count =  1    |0|  selectionSequenceId = 301  |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |       Scope 1 Length = 4      |0| subTemplateMultiList =  ZZZ |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |     Field Length = 0xFFFF     |0|      selectorId = 302       |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |        Field Length = 4       |0|      selectorId = 302       |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |        Field Length = 4       |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure W: Options Template Record for PSAMP SSRI using subTemplateMultiList

A subTemplateMultiList with semantic allOf is used to encode the list of Observation Points.

```
   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 = 2           |          Length = 16          |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |        Template ID = 263      |         Field Count = 2       |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |0|   exporterIPv4Address = 8   |        Field Length = 4       |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |0|   ingressInterface = 10     |        Field Length = 4       |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure X: PSAMP SSRI, Template Record for interface

```
 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 = 2           |          Length = 16          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Template ID = 264      |        Field Count = 2        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|   exporterIPv4Address = 8   |        Field Length = 4       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|      lineCardId = 141       |        Field Length = 4       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure Y: PSAMP SSRI, Template Record for linecard

```
 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 = 2           |          Length = 20          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Template ID = 265      |        Field Count = 3        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|   exporterIPv4Address = 8   |      Field Length = 4         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|      lineCardId = 141       |       Field Length = 4        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|    ingressInterface = 10    |       Field Length = 4        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure Z: PSAMP SSRI, Template Record for linecard and interface

The PSAMP SSRI Data Set is represented 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 = 262         |           Length = 68         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    selectionSequenceId = 7                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      255      | Observation Point List Len=49 |semantic=allOf |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
 |      OP1 Template ID = 263     |        OP1 Length = 12        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |          Router 1 exporterIPv4Address = 192.0.2.11             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                   OP1 ingressInterface = 1                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |   OP2&OP3 Template ID = 264    |    OP2 & OP3 Length = 20      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |          Router 2 exporterIPv4Address = 192.0.2.12             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                       OP2 lineCardId = A                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |          Router 3 exporterIPv4Address = 192.0.2.13             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                       OP3 lineCardId = B                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |      OP4 Template ID = 265     |         OP4 Length = 16       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |          Router 4 exporterIPv4Address = 192.0.2.14             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                       OP4 lineCardId = C                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                    OP4 ingressInterface = 2                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                          selectorId = 5                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                          selectorId = 10                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure ZA: Example of a PSAMP SSRI Data Record, Encoded using a subTemplateMultiList

Note that the Data Record above contains multiple instances of Template 264 to represent Observation Point 2 (Router2, line card A) and Observation Point 3 (Router3, line card B). Instead, if a single Observation Point had both line card A and line card B, a basicList would be used to represent the list of line cards.

9. Relationship with the Other IFPIX Documents

9.1. Relationship with Reducing Redundancy

"Reducing Redundancy in IP Flow Information Export (IPFIX) and Packet Sampling (PSAMP) Reports" [RFC5473] describes a bandwidth saving method for exporting Flow or packet information using the IP Flow Information eXport (IPFIX) protocol.

It defines the commonPropertiesID Information Element for exporting Common Properties.

9.1.1. Encoding Structured Data Element using Common Properties.

When Structured Data Information Elements contain repeated elements, these elements may be replaced with a commonPropertiesID Information Element as specified in [RFC5473]. The replaced elements may include the basicList, subTemplateList and subTemplateMultiList Information Elements.

This technique might help reducing the bandwidth requirements for the export. However, a detailed analysis of the gain has not been done; refer to Section 8.3 of [RFC5473] for further considerations.

9.1.2. Encoding Common Properties elements With Structured Data Information Element.

Structured Data Information Element MAY be used to define a list of commonPropertiesID, as a replacement for the specifications in [RFC5473].

Indeed, the example in figures 1 and 2 of [RFC5473] can be encoded with the specifications in this document.

```
+----------------+-------------+---------------------------+
| sourceAddressA | sourcePortA |     <Flow1 information>   |
+----------------+-------------+---------------------------+
| sourceAddressA | sourcePortA |     <Flow2 information>   |
+----------------+-------------+---------------------------+
| sourceAddressA | sourcePortA |     <Flow3 information>   |
+----------------+-------------+---------------------------+
| sourceAddressA | sourcePortA |     <Flow4 information>   |
+----------------+-------------+---------------------------+
|      ...       |     ...     |            ...            |
+----------------+-------------+---------------------------+
```

Figure ZB: Common and Specific Properties Exported Together [RFC5473]

```
+------------------------+-----------------+-------------+
| index for properties A | sourceAddressA  | sourcePortA |
+------------------------+-----------------+-------------+
|          ...           |      ...        |     ...     |
+------------------------+-----------------+-------------+


+------------------------+---------------------------+
| index for properties A |     <Flow1 information>   |
+------------------------+---------------------------+
| index for properties A |     <Flow2 information>   |
+------------------------+---------------------------+
| index for properties A |     <Flow3 information>   |
+------------------------+---------------------------+
| index for properties A |     <Flow4 information>   |
+------------------------+---------------------------+
```

Figure ZC: Common and Specific Properties Exported Separately according to [RFC5473]

```
+----------------+-------------+---------------------------+
| sourceAddressA | sourcePortA |     <Flow1 information>   |
+----------------+-------------+---------------------------+
                               |     <Flow2 information>   |
                               +---------------------------+
                               |     <Flow3 information>   |
                               +---------------------------+
                               |     <Flow4 information>   |
                               +---------------------------+
                               |            ...            |
                               +---------------------------+
```

Figure ZD: Common and Specific Properties Exported with Structured Data Information Element

The example in figure ZD could be encoded with a basicList if the <Flow information> represents a single Information Element, with a subTemplateList if the <Flow information> represents a Template Record, or with a subTemplateMultiList if the <Flow information> is composed of different Template Records.

Using Structured Data Information Elements as a replacement for the techniques specified in "Reducing Redundancy in IP Flow Information Export (IPFIX) and Packet Sampling (PSAMP) Reports" [RFC5473] offers the advantage that a single Template Record is defined. Hence the Collectors job is simplified in terms of Template management and combining Template/Options Template Records.

However, it must be noted that using Structured Data Information Elements as a replacement for the techniques specified in "Reducing Redundancy in IP Flow Information Export (IPFIX) and Packet Sampling (PSAMP) Reports" only applies to simplified cases. For example, the "Multiple Data Reduction" (Section 7.1 [RFC5473]) might be too complex to encode with Structured Data Information Elements.

9.2. Relationship with Guidelines for IPFIX Testing

[RFC5471] presents a list of tests for implementers of IP Flow Information eXport (IPFIX) compliant Exporting Processes and Collecting Processes.

Although [RFC5471] doesn't define any structured data element specific tests, the Structured Data Information Elements can be used in many of the [RFC5471] tests.

The [RFC5471] series of test could be useful because the document specifies that every Information Element type should be tested. However, not all cases from this document are tested in [RFC5471].

The following sections are especially noteworthy:

- 3.2.1. Transmission of Template with fixed size Information Elements

  - each data type should be used in at least one test. The new data types specified in Section 4.1. should be included in this test.

- 3.2.2. Transmission of Template with variable length Information Elements

  - this test should be expanded to include Data Records containing variable length basicList, subTemplateList, and subTemplateMultiList Information Elements.

. 3.3.1. Enterprise-specific Information Elements

- this test should include the export of basicList, subTemplateList, and subTemplateMultiList Information Elements containing Enterprise-specific Information Elements. e.g., see the example in figure B.

. 3.3.3. Multiple instances of the same Information Element in one Template

- this test should verify that multiple instances of the basicList, subTemplateList and subTemplateMultiList Information Elements are accepted.

. 3.5 Stress/Load tests

- since the structured data types defined here allow modeling of complex data structures, they may be useful for stress testing both Exporting Processes and Collecting Processes.

9.3. Relationship with Bidirectional Flow Export

[RFC5103] describes a method for exporting bidirectional flow information, and defines the biflowDirection Information Element for this purpose.

[RFC5103] Biflows may be encoded in a subTemplateList or subTemplateMultiList. The basicList requires recurrence of a single element, so is not suitable for Biflows.

Encoding Biflows with subTemplateList or subTemplateMultiList provides a more logical division of the information in both directions, although this encoding incurs a small additional bandwidth penalty.

An example of Biflow encoding using Structure Data Information Elements and comparison with the [RFC5103] Biflow encoding is shown in Appendix B.

9.4. Relationship with IPFIX Mediation Function

The Structured Data Information Elements would be beneficial for the export of aggregated Data Records in mediation function, as was demonstrated with the example of the aggregated Observation Point in Section 5.3.

10. IANA Considerations

This document specifies several new IPFIX abstract data types, a new IPFIX Data Type Semantic, and several new Information Elements.

These require the creation of two new IPFIX registries and updating the existing IPFIX Information Element registry as detailed below.

10.1. New Abstract Data Types

Section 4.1. of this document specifies several new IPFIX abstract data types. Per Section 6 of the IPFIX information model [RFC5102], new abstract data types can be added to the IPFIX information model. This requires creation of a new IPFIX "abstract data types" registry at http://www.iana.org/assignments/ipfix. This registry should include all the abstract data types from Section 3.1 of [RFC5102].

Abstract data types to be added to the IPFIX "abstract data types" registry are listed below.

10.1.1. basicList

The type "basicList" represents a list of any Information Element used for single-valued data types.

10.1.2. subTemplateList

The type "subTemplateList" represents a list of a structured data type, where the data type of each list element is the same and corresponds with a single Template Record.

10.1.3. subTemplateMultiList

The type "subTemplateMultiList" represents a list of structured data types, where the data types of the list elements can be different and correspond with different template definitions.

10.2. New Data Type Semantics

Section 4.2. of this document specifies a new IPFIX Data Type Semantic. Per Section 3.2 of the IPFIX information model [RFC5102], new data type semantics can be added to the IPFIX information model. Therefore, the IANA IPFIX informationElementSemantics registry [IANA-IPFIX], which contains all the data type semantics from Section 3.2 of [RFC5102], must be augmented with the "list" value below.

10.2.1. list

A list is a structured data type, being composed of a sequence of elements e.g. Information Element, Template Record, etc.

10.3. New Information Elements

Section 4.3. of this document specifies several new Information Elements which are to be created in the IPFIX Information Element registry [IANA-IPFIX].

New Information Elements to be added to the IPFIX Information Element registry are listed below.

EDITOR’S NOTE: the XML specification in Appendix A must be updated with the elementID values allocated below.

10.3.1. basicList

Name: basicList
Description:
Specifies a generic Information Element with a basicList abstract data type. For example, a list of port numbers, a list of interface indexes, etc.
Abstract Data Type: basicList
Data Type Semantics: list
ElementId: XXX (to be specified by IANA)
Status: current

10.3.2. subTemplateList

Name: subTemplateList
Description:
Specifies a generic Information Element with a subTemplateList abstract data type.
Abstract Data Type: subTemplateList
Data Type Semantics: list
ElementId: YYY (to be specified by IANA)
Status: current

10.3.3. subTemplateMultiList

Name: subTemplateMultiList
Description:
Specifies a generic Information Element with asubTemplateMultiList abstract data type.
Abstract Data Type: subTemplateMultiList
Data Type Semantics: list
ElementId: ZZZ (to be specified by IANA)
Status: current

10.4. New Structured Data Semantics

Section 4.4. of this document specifies a series of new IPFIX structured data type semantics, which is expressed as an 8-bit value. This requires the creation of a new IPFIX "structured data types semantics" IPFIX subregistry [IANA-IPFIX].

Entries may be added to this subregistry subject to a Standards Action [RFC5226]. Initially, this registry should include all the structured data type semantics listed below.

10.4.1. undefined

Name: undefined

Description: The "undefined" structured data type semantic specifies that the semantic of list elements is not specified, and that, if a semantic exists, then it is up to the Collecting Process to draw its own conclusions. The "undefined" structured data type semantic is the default structured data type semantic.

Value: 0xFF

Reference: <this future RFC>

10.4.2. noneOf

Name: noneOf

Description: The "noneOf" structured data type semantic specifies that none of the elements are actual properties of the Data Record.

Value: 0x00

Reference: <this future RFC>

10.4.3. exactlyOneOf

Name: exactlyOneOf

Description: The "exactlyOneOf" structured data type semantic specifies that only a single element from the structured data is an actual property of the Data Record. This is equivalent to a logical XOR operation.

Value: 0x01

Reference: <this future RFC>

10.4.4. oneOrMoreOf

Name: oneOrMoreOf

Description: The "oneOrMoreOf" structured data type semantic specifies that one or more elements from the list in the structured data are actual properties of the Data Record. This is equivalent to a logical OR operation.

Value: 0x02

Reference: <this future RFC>

10.4.5. allOf

Name: allOf

Description: The "allOf" structured data type semantic specifies that all of the list elements from the structured data are actual properties of the Data Record.

Value: 0x03

Reference: <this future RFC>

10.4.6. ordered

Name: ordered

Description: The "ordered" structured data type semantic specifies that elements from the list in the structured data are ordered.

Value: 0x04

Reference: <this future RFC>

11. Security Considerations

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

12. References

12.1. 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] T. Narten, T., Alverstrand, H. , "Guidelines for Writing an IANA Considerations Section in RFCs", RFC5226, May 2008.

12.2. Informative References

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

[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.

[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., Ed., "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.

[IANA-IPFIX] http://www.iana.org/assignments/ipfix/ipfix.xhtml

13. Acknowledgement

The authors would like to thank Zhipu Jin, Nagaraj Varadharajan, Brian Trammel, Atsushi Kobayashi, Rahul Patel for their feedback, and Gerhard Muenz, for proof reading the document.

14. Authors' Addresses

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

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

Gowri Dhandapani
Cisco Systems Inc.
13615 Dulles Technology Drive
Herndon, Virigina 20171
United States

Phone: +1 408 853 0480
EMail: gowri@cisco.com

Stan Yates
Cisco Systems Inc.
7100-8 Kit Creek Road
PO Box 14987
Research Triangle Park
North Carolina, 27709-4987
United States

Phone: +1 919 392 8044
EMail: syates@cisco.com

Paul Aitken
Cisco Systems (Scotland) Ltd.
96 Commercial Quay
Commercial Street
Edinburgh, EH6 6LX, United Kingdom

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

Appendix A. Additions to XML Specification of IPFIX Information Elements and Abstract Data Types

This appendix 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 4. (generated from this appendix) is normative.

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

```
<field name="basicList"
        dataType="basicList"
        group="structured-data"
        dataTypeSemantics="List"
        elementId="XXX" applicability="all" status="current">
   <description>
     <paragraph>
        Represents a list of zero or more instances of
        any Information Element, primarily used for
        single-valued data types. For example, a list of port
        numbers, list of interface indexes, list of AS in a
        BGP AS-PATH, etc.
     </paragraph>
   </description>
 </field>

 <field name="subTemplateList"
        dataType="subTemplateList"
        group="structured-data"
        dataTypeSemantics="List"
        elementId="YYY" applicability="all" status="current">
   <description>
     <paragraph>
        Represents a list of zero or more instances of a
        structured data type, where the data type of each list
        element is the same and corresponds with a single
        Template Record. For example, a structured data type
        composed of multiple pairs of ("MPLS label stack entry
        position", "MPLS label stack value"), a structured data
        type composed of performance metrics, a structured data
        type composed of multiple pairs of IP address, etc.
     </paragraph>
   </description>
```

```
     </field>

     <field name="subTemplateMultiList"
            dataType="subTemplateMultiList"
            group="structured-data"
            dataTypeSemantics="List"
            elementId="ZZZ" applicability="all" status="current">
       <description>
         <paragraph>
           Represents a list of zero or more instances of
           structured data types, where the data type of each list
           element can be different and corresponds with
           different template definitions. For example, a
           structured data type composed of multiple access-list
           entries, where entries can be composed of different
           criteria types.
         </paragraph>
       </description>
     </field>
```

The following structured data type semantic definitions are appended to the the IPFIX information model in Appendix A of [RFC5102].

```
   <structuredDataTypeSemantics>
     <structuredDataTypeSemantic name="undefined" value="255">
       <description>
         <paragraph>
          The "undefined" structured data type semantic specifies
          that the semantic of list elements is not specified, and
          that, if a semantic exists, then it is up to the
          Collecting Process to draw its own conclusions.  The
          "undefined" structured data type semantic is the default
          structured data type semantic.
         </paragraph>
       </description>
     </structuredDataTypeSemantic>

     <structuredDataTypeSemantic name="noneOf" value="0">
       <description>
         <paragraph>
          The "noneOf" structured data type semantic specifies
          that none of the elements are actual properties of the
          Data Record.
         </paragraph>
```

```
        </description>
      </structuredDataTypeSemantic>

      <structuredDataTypeSemantic name="exactlyOneOf" value="1">
        <description>
          <paragraph>
           The "exactlyOneOf" structured data type semantic
           specifies that only a single element from the structured
           data is an actual property of the Data Record.  This is
           equivalent to a logical XOR operation.
          </paragraph>
        </description>
      </structuredDataTypeSemantic>

      <structuredDataTypeSemantic name="oneOrMoreOf" value="2">
        <description>
          <paragraph>
           The "oneOrMoreOf" structured data type semantic
           specifies that one or more elements from the list in the
           structured data are actual properties of the Data
           Record.  This is equivalent to a logical OR operation.
          </paragraph>
        </description>
      </structuredDataTypeSemantic>

      <structuredDataTypeSemantic name="allOf" value="3">
        <description>
          <paragraph>
           The "allOf" structured data type semantic specifies that
           all of the list elements from the structured data are
           actual properties of the Data Record.
          </paragraph>
        </description>
      </structuredDataTypeSemantic>

      <structuredDataTypeSemantic name="ordered" value="4">
        <description>
          <paragraph>
           The "ordered" structured data type semantic specifies
           that elements from the list in the structured data are
           ordered.
          </paragraph>
        </description>
      </structuredDataTypeSemantic>
    </structuredDataTypeSemantics>
```

The following schema definitions are appended to the abstract data types defined in Appendix B of [RFC5102].

```
<simpleType name="dataType">
  <restriction base="string">
    <enumeration value="basicList">
      <annotation>
        <documentation>
          Represents a list of zero or more instances of
          any Information Element, primarily used for
          single-valued data types. For example, a list of port
          numbers, list of interface indexes, list of AS in a
          BGP AS-PATH, etc.
        </documentation>
      </annotation>
    </enumeration>
    <enumeration value="subTemplateList">
      <annotation>
        <documentation>
          Represents a list of zero or more instances of a
          structured data type, where the data type of each list
          element is the same and corresponds with a single
          Template Record. For example, a structured data type
          composed of multiple pairs of ("MPLS label stack entry
          position", "MPLS label stack value"), a structured
          data type composed of performance metrics, a
          structured data type composed of multiple pairs of IP
          address, etc.
        </documentation>
      </annotation>
    </enumeration>
    <enumeration value="subTemplateMultiList">
      <annotation>
        <documentation>
          Represents a list of zero or more instances of
          structured data types, where the data type of each
          list element can be different and corresponds with
          different template definitions. For example, a
          structured data type composed of multiple
          access-list entries, where entries can be
          composed of different criteria types.
        </documentation>
      </annotation>
    </enumeration>
  </restriction>
</simpleType>
```

```
   <simpleType name="dataTypeSemantics">
     <restriction base="string">
       <enumeration value="List">
         <annotation>
           <documentation>
             Represents an arbitrary-length sequence of structured
             data elements, either composed of regular Information
             Elements or composed of data conforming to a Template
             Record.
           </documentation>
         </annotation>
       </enumeration>
     </restriction>
   </simpleType>

   <complexType name="structuredDataTypeSemantics">
     <sequence>
       <element name="structuredDataTypeSemantic"
                minOccurs="1" maxOccurs="unbounded">
         <complexType>
           <sequence>
             <element name="description" type="text"/>
           </sequence>
           <attribute name="name" type="string" use="required"/>
           <attribute name="value" type="unsignedByte"
 use="required"/>
         </complexType>
       </element>
     </sequence>
   </complexType>

   <element name="structuredDataTypeSemantics"
            type="structuredDataTypeSemantics">
     <annotation>
       <documentation>
         structured data type semantics express the relationship
         among multiple list elements in a structured data
         Information Element.
       </documentation>
     </annotation>
   </element>
```

Appendix B. Example of Biflow Encoding using Structured Data Information Elements

Referring to [RFC5103] figure 1, a Biflow consists of two parts: some "key" fields such as src/dst information (IP addresses, ports), followed by a set of forward/reverse pairs.

Then looking at [RFC5103] figure 7, we see that the Reverse PEN is repeated many times to indicate fields which were observed in the reverse direction.

Looking back at [RFC5103] figure 1, it's clear that the encoding can use a Template Record consisting of the Flow Keys followed by a subTemplateList consisting of two elements: one for the forward direction, the other for the reverse direction.

The subTemplateList uses a single Template Record to describe the fields in both lists since they are a set of forward/reverse pairs.

```
              Uniflow                              Uniflow
+-------+-------+-----------------+ +-------+-------+-----------------+
| src A | dst B | counters/values | | src B | dst A | counters/values |
+-------+-------+-----------------+ +-------+-------+-----------------+
    |       |            |                              |
    V       V            V                              V
                  <----------- subTemplateList -------------->
+-------+-------+---------------------+---------------------+
| src A | dst B | fwd counters/values | rev counters/values |
+-------+-------+---------------------+---------------------+
         |                  |                    |
         V                  V                    V
    key fields         fwd element          rev element
```

Figure B0: Using a subTemplateList to represent a Biflow.

The following example shows the example from Appendix A of [RFC5103] encoded using a subTemplateList:

```
  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 = 2           |          Length = 24          |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |      Template ID = 266        |        Field Count = 4        |
```

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0| flowDirection            61 |       Field Length = 1        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0| flowStartSeconds        150 |       Field Length = 4        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0| octetTotalCount          85 |       Field Length = 4        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0| packetTotalCount         86 |       Field Length = 4        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure B1: Template for the Biflow Fields

```
 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 = 2           |          Length = 32          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Template ID = 267        |        Field Count = 6        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0| sourceIPv4Address         8 |       Field Length = 4        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0| destinationIPv4Address   12 |       Field Length = 4        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0| sourceTransportPort       7 |       Field Length = 2        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0| destinationTransportPort 11 |       Field Length = 2        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0| protocolIdentifier        4 |       Field Length = 1        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|  subTemplateList = YYY      |       Field Length = 29       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure B2: Template for the Key Fields

A subTemplateList with semantic allOf is used for encoding the BiFlow fields for the forward and reverse directions. Note that the subTemplateList is encoded using fixed length, as shown in the above Template definition.

Also, note that the overall template size is 24 + 32 = 56 octets, compared with 64 octets in the [RFC5103] example - so a small saving is achieved for the Template Record.

```
 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 = 267            |          Length = 46          |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                 sourceIPv4Address = 192.0.2.2                 |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |              destinationIPv4Address = 192.0.2.3               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |  sourceTransportPort = 32770  | destinationTransportPort = 80 |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | protocol = 6  |semantic=allOf |       Template ID = 266       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | dir = forward |     flowStartSeconds = 2006-02-01 17:00:00    |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |      ...      |           octetTotalCount = 18000             |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |      ...      |           packetTotalCount = 65               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |      ...      | dir = reverse |   flowStartSeconds  = ...     |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |  ...    2006-02-01 17:00:01   |  octetTotalCount = 128000 ... |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |              ...              |   packetTotalCount = 110      |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |              ...              |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure B3: Biflow Data Set Encoded using Structured Data

Note that the Data Set length is 46, compared with 41 in RFC5103. The five additional octets are due to the inclusion of the 8-bit semantic, 16-bit Template ID and two, 8-bit direction indicators.

Clearly structured data offers an alternative way to encode Biflows, without the private Reverse PEN specified in [RFC5103].

Appendix C. Encoding IPS Alert using Structured Data Information Elements

In this section, an IPS alert example is used to demonstrate how complex data and multiple levels of hierarchy can be encoded using Structured Data Information Elements. Also, this example demonstrates how a basicList of subTemplateLists can be used to represent semantics at multiple levels in the hierarchy.

An IPS alert consists of the following mandatory attributes: signatureId, protocolIdentifier and riskRating. It can also contain zero or more participants, each participant can contain zero or more attackers and zero or more targets. An attacker contains the attributes sourceIPv4Address and applicationId, and a target contains the attributes destinationIPv4Address and applicationId.

Note that the signatureId and riskRating Information Element fields are created for these examples only; the Field IDs are shown as N/A. The signatureId helps to uniquely identify the IPS signature that triggered the alert. The riskRating identifies the potential risk, on a scale of 0-100 (100 being most serious), of the traffic that triggered the alert.

Consider the example described in case study 2 of Section 5.6. The IPS alert contains participants encoded as a subTemplateList with semantic allOf. Each participant uses a basicList of subTemplateLists to represent attackers and targets. For the sake of simplicity, the alert has two participants P1 and P2. In participant P1, attacker A1 or A2 attack target T1. In participant P2, attacker A3 attacks targets T2 and T3.

```
Participant P1:

     (basicList, allof,

           (subTemplateList, exactlyOneOf, attacker A1, A2)

           (subTemplateList, undefined, target T1)

     )

Participant P2:

     (basicList, allOf,

           (subTemplateList, undefined, attacker A3,

           (subTemplateList, allOf, targets T2, T3)

     )

Alert :

        (subTemplateList, allOf, Participant P1, Participant P2)
```

```
-------------------------------------------------------------------
      |        |        |             participant
sigId |protocol| risk   |      attacker    |      target
      |   Id   | Rating |    IP    | appId |    IP      | appId
-------------------------------------------------------------------
1003     17      10      192.0.2.3  103    192.0.2.103    3001
                         192.0.2.4  104

                         192.0.2.5  105    192.0.2.104    4001
                                           192.0.2.105    5001
-------------------------------------------------------------------
```

Participant P1 contains:
Attacker A1: (IP, appID)=(192.0.2.3, 103)
Attacker A2: (IP, appID)=(192.0.2.4, 104)
Target T1: (IP, appID)= (192.0.2.103, 3001)

Participant P2 contains:
Attacker A3: (IP, appID) = (192.0.2.5, 105)
Target T2: (IP, appID)= (192.0.2.104, 4001)
Target T3: (IP, appID)= (192.0.2.105, 5001)

To represent an alert, the following Templates are defined:
Template for target (268)
Template for attacker (269)
Template for participant (270)
Template for alert (271)

```
     alert (271)
     |  (signatureId)
     |  (protocolIdentifier)
     |  (riskRating)
     |
     +------- participant (270)
              |
              +------- attacker (269)
              |           (sourceIPv4Address)
              |           (applicationId)
              |
              +------- target (268)
                       |  (destinationIPv4Address)
                       |  (applicationId)
```

Note that the attackers are always composed of a single
applicationId, while the targets typically have multiple

applicationId, for the sake of simplicity this example shows only one applicationId in the target.

Template Record for target, with the Template ID 268:

```
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 = 2            |      Length = 16 octets       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Template ID = 268      |       Field Count = 2         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0| destinationIPv4Address = 12 |       Field Length = 4        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|       applicationId = 95    |       Field Length = 4        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure C0: Encoding IPS Alert, Template for Target

Template Record for attacker, with the Template ID 269:

```
 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 = 2           |      Length = 16 octets       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Template ID = 269      |       Field Count = 2         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|    sourceIPv4Address = 8    |       Field Length = 4        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|     applicationId = 95      |       Field Length = 4        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure C1: Encoding IPS Alert, Template for Attacker

Template Record for participant, with the Template ID 270:

```
 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 = 2           |      Length = 12 octets       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Template ID = 270      |       Field Count = 1         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
 |0|       basicList = XXX       |     Field Length = 0xFFFF     |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure C2: Encoding IPS Alert, Template for Participant

The Template Record for the participant has one basicList Information Element, which is a list of subTemplateLists of attackers and targets.

Template Record for IPS alert, with the Template ID 271:

```
  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 = 2            |      Length = 24 octets       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |       Template ID = 271       |       Field Count = 4         |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0|    signatureId = N/A        |       Field Length = 2        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0|   protocolIdentifier = 4    |       Field Length = 1        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0|     riskRating = N/A        |       Field Length = 1        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0|     subTemplateList = YYY   |     Field Length = 0xFFFF     |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure C3: Encoding IPS Alert, Template for IPS Alert

The subTemplateList in the alert Template Record contains a list of participants.

The Length of basicList and subTemplateList are encoded in three bytes even though they may be less than 255 octets.

The Data Set is represented 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 = 271         |         Length = 102          |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |      signatureId = 1003       | protocolId=17 | riskRating=10 |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
 |      255      |participant List Length  = 91  |semantic=allOf |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | participant Template ID = 270 |     255       | P1 List Len = |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |      41       | semantic=allOf|    P1 List Field ID = YYY     |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | P1 List Field ID Len = 0xFFFF |      255      |P1 attacker ...|
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | List Len = 19 |sem=exactlyOne | P1 attacker Template ID = 269 |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |          P1 attacker A1 sourceIPv4Address = 192.0.2.3         |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |               P1 attacker A1 applicationId = 103              |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |          P1 attacker A2 sourceIPv4Address = 192.0.2.4         |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |               P1 attacker A2 applicationId = 104              |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |      255      | P1 target List Len = 11       | sem=undefined |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |  P1 target Template ID = 268  | P1 target T1 destinationIPv4  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | ... Address = 192.0.2.103     |P1 target T1 applicationId =...|
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | ...       3001                |      255      | P2 List Len = |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | ...  41       | semantic=allOf|    P2 List Field ID = YYY     |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | P2 List Field ID Len = 0xFFFF |      255      |P2 attacker ...|
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | List Len = 11 | sem=undefined | P2 attacker Template ID = 269 |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |          P2 attacker A3 sourceIPv4Address = 192.0.2.5         |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |               P2 attacker A3 applicationId = 105              |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |      255      |    P2 target List Len = 19    |semantic=allOf |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |  P2 target Template ID = 268  | P2 target T2 destinationIPv4  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | ... Address = 192.0.2.104     |P2 target T2 applicationId =...|
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | ...       4001                | P2 target T3 destinationIPv4  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | ... Address = 192.0.2.105     |P2 target T3 applicationId =...|
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | ...       5001                |
```

```
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure C4: Encoding IPS Alert, Data Set

IPFIX Working Group
Internet-Draft
Intended Status: Standards Track
Expires: April 16, 2011
A. Johnson
B. Claise
P. Aitken
Cisco Systems, Inc.
October 16, 2010

Exporting MIB variables using the IPFIX Protocol
draft-johnson-ipfix-mib-variable-export-00

Abstract

This document specifies a way to export Management Information Base (MIB) objects within the IPFIX protocol, avoiding the need to define new IPFIX Information Elements for existing Management Information Base objects that are already fully specified.

This method requires an extension to the current IPFIX protocol. New Template Set and Options Template Sets are specified to allow the export of Simple Network Management Protocol (SNMP) MIB Objects.

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.. 4
2. Terminology... 6
3. Example Use Cases................................... 6
3.1 Detailing CPU Load History.......................... 6
3.2 Output Interface Queue Size in PSAMP Packet Report 7
4. MIB OID Extended Template Formats................... 8
4.1 MIB OID Extended Template Record Format........... 8
4.2 MIB OID Extended Options Template Record Format... 9
4.3 MIB OID Extended Field Specifier Format.......... 11
4.3.1 Standard Field Specifier Format................. 11
4.3.2 Extended Field Specifier Format for non-indexed MIB Object 12
4.3.3 Extended Field Specifier Format for Indexed MIB Object 15
4.4 Template Management............................. 20
5. The Collecting Process's Side.................... 21
6. Applicability.................................... 21
7. Security Considerations.......................... 22
8. IANA Considerations.............................. 22
9. References....................................... 23
9.1 Normative References............................ 23
9.2 Informative References.......................... 23

TO DO

ACTION: need a third example with an Options Template Set to show TBD2 in action. TBD2 is the Set ID of an Options Template Set that uses the extended Field Specifier.

1. Introduction

There is growing interest in using IPFIX to export management information, especially since periodically exporting large chunks of repetitive data from a device is often more appropriate than using a polling mechanism.

While initially targeted at different problems, there is a large parallel between the information transported via IPFIX and SNMP.

Furthermore, certain Management Information Base objects are highly relevant to flows as they are understood today.

For example, in the IPFIX information model [RFC5102], several Information Elements coming from the SNMP world have already been specified. e.g. the interface's SNMP index (ifIndex, [RFC2863]), the interface's name (ifName,[RFC2863]) and the system uptime (sysUpTime, [RFC3418]). Rather than map existing SNMP MIB Object Identifiers to IPFIX Information Elements on a case by case basis, causing replication between the two models, it would be advantageous to enable the export of any existing or future SNMP objects as part of an IPFIX Data Record. However, this is not possible using the existing IPFIX Template Sets as specified in [RFC5101].

Another advantage of exporting MIB objects via IPFIX is that IPFIX would benefit from an extended series of types to be exported. Indeed, the simple and application-wide data types specified in SMIv2 [RFC2578], along with a new textual conventions, can be exported within IPFIX and then decoded in the Collector.

Yet another advantage of exporting MIB objects via IPFIX is the synchronization of the Data Record counters and the MIB counters. For example, if a Network Management Station (NMS) needs the interface counters, coming from ifTable in "The Interfaces Group MIB" [RFC2863]), at the time the Data Record ends, the NMS must poll the interface counters after receiving the Data Records. Unless synchronized Data Records export and SNMP polling is in place (which is difficult/impossible in practice because the Flow duration can not be predicted), the Flow counters and interface counters cannot be compared. With the export of the interface counters within the Data Record, this problem is avoided.

In this document, new Template Sets for Data Records and Options Records are specified to allow Templates to contain any combination of fields defined by traditional IPFIX Information Elements and/or MIB Object Identifiers. The MIB Object Identifiers can reference either non-indexed or indexed MIB objects. When an indexed MIB object is exported, a method to identify how that MIB object was indexed is specified so that the full meaning of the information being exported can be conveyed. A set of example use cases is used to illustrate how these specifications can be used.

Since IPFIX is a push mechanism, initiated from the Exporter with no acknowledgment method, this specification doesn't provide the ability to execute configuration, unlike the SNMP protocol. Instead, this specification allows adding the value of MIB objects into IPFIX Data Records.

2. Terminology

IPFIX-specific terminology used in this document is defined in section 2 of [RFC5101]. For example: Information Element, Template, Template Record, Options Template Record, Template Set, Collector, Exporter, Flow Record, etc... As in [RFC5101], these IPFIX-specific terms have the first letter of a word capitalized.

This document prefers the more generic term "Data Record" as opposed to "Flow Record" as this specification allows the export of MIB objects.

MIB Object Identifier (MIB OID)

> AlphaNumeric-format variable name, denoting a variable name expressed as a sequence of decimal numbers or names separated by periods, as specified by the OBJECT IDENTIFIER in [RFC2578].

MIB Object Identifier Information Element

> An IPFIX Information Element ("MIBObjectIdentifierMarker") that denotes that a MIB Object Identifier is exported in the (Options) Template Record.

3. Example Use Cases

3.1 Detailing CPU Load History

The CPU Usage of a remote network device could be monitored by configuring it to periodically send CPU usage information to a centralized Collector. In this example, the Exporter would send an IPFIX Message every 30 minutes that contained Data Records detailing the CPU 1 minute busy average at 1 minute intervals.

The table of data that is to be exported would look like:

TIMESTAMP	CPU BUSY PERCENTAGE
StartTime + 0 seconds	10%
StartTime + 60 seconds	14%
StartTime + 120 seconds	19%
StartTime + 180 seconds	16%
StartTime + 240 seconds	23%
StartTime + 300 seconds	29%
...	...

The Template Record for such a Data Record will detail two Information Elements:

flowStartSeconds from [RFC5102]:
  Value: IE = 150
  Description: The absolute timestamp of the first packet of this Flow.

cpmCPUTotal1minRev from the proprietary CISCO-PROCESS-MIB
  Value: MIB OID = "1.3.6.1.4.1.9.9.109.1.1.1.1.7"
  Description: The overall CPU busy percentage in the last one-minute period

3.2 Output Interface Queue Size in PSAMP Packet Report

If a PSAMP Packet Report [RFC5476] was generated on any dropped packets on an interface then it may be desirable to know if the send queue on the output interface was full. This could be done be sending the size of the send queue in the same Data Record as the PSAMP Packet Report.

The exported data could look something like:

SRC ADDR	DST ADDR	PAK LEN	OUTPUT I/F	OUTPUT Q. SIZE
192.0.2.1	192.0.2.3	150	Eth 1/0	45
192.0.2.4	192.0.2.9	350	Eth 1/0	45
192.0.2.3	192.0.2.9	650	Eth 1/0	23
192.0.2.4	192.0.2.6	350	Eth 1/1	0

The MIB object for the Output Queue Length, ifOutQLen ("1.3.6.1.2.1.2.2.1.21"), is indexed by the ifIndex interface index as detailed in the IF-MIB [RFC2863]. If, for example, the interface index of "Eth 1/0" in our example is 15, the full MIB Object Identifier for the Output Queue Length would be

"1.3.6.1.2.1.2.2.1.21.15". With the specification in this document, each time a different MIB OID is specified in the Template Record, a new MIB object must be identified, hence a new Template Record. Rather than send a separate Template Record for each Interface Index, it would be much more convenient to identify the index in the Data Record itself.

In fact, only how the indexed object was indexed is important. In our example we identify the Egress Interface, although for other uses it may be sufficient to know that the Output Queue Size was taken from the interface that the packet was switched out of without identifying the actual interface.

The Template Record for our example Data Record would contain the following Information Elements:

```
sourceIPv4Address
destinationIPv4Address
totalLengthIPv4
egressInterface
outboundQueueLength indexed by: egressInterface
```

4. MIB OID Extended Template Formats

Extended Template Record Formats are required to send data defined by MIB Object Identifiers. New Template Sets are required for these extended Template Record Formats.

4.1 MIB OID Extended Template Record Format

The format of the MIB Object Identifier Extended Template Record is shown in Figure A. It consists of a Template Record Header and one or more Field Specifiers.

```
+--------------------------------------------------+
| Template Record Header                           |
+--------------------------------------------------+
| Field Specifier                                  |
+--------------------------------------------------+
| Field Specifier                                  |
+--------------------------------------------------+
...
+--------------------------------------------------+
| Field Specifier                                  |
+--------------------------------------------------+
```

Figure A: MIB Object Identifier Extended Template Record Format

A MIB Object Identifier Extended Template Record MUST contain at least one MIB Object Identifier Extended Field Specifier. It MAY also contain any combination of IANA-assigned and/or Enterprise-Specific Information Element identifiers as specified in [RFC5101].

The format of the Template Record Header is shown in Figure B.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Template ID (> 255)   |         Field Count           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure B: Template Record Header Format

Where:

Template ID

Template ID of this Template Record. This value is greater than 255.

Field Count

Number of all fields in this Template Record, including the Scope Fields.

At this level of detail the layout of the Template Record Format, as specified in [RFC5101], and the MIB Object Identifier Extended Template Record Format are identical. It is only the structure of the Field Specifiers that is different (see section 4.3).

4.2 MIB OID Extended Options Template Record Format

The format of the MIB Object Identifier Extended Options Template Record is shown in Figure C. It consists of an Options Template Record Header and one or more Field Specifiers.

```
        +--------------------------------------------------+
        | Options Template Record Header                   |
        +--------------------------------------------------+
        | Field Specifier                                  |
        +--------------------------------------------------+
        | Field Specifier                                  |
        +--------------------------------------------------+
                                 ...
        +--------------------------------------------------+
        | Field Specifier                                  |
        +--------------------------------------------------+
```

Figure C: MIB Object Identifier Options Extended Template Record Format

A MIB Object Identifier Extended Options Template Record MUST contain at least one MIB Object Identifier Extended Field Specifier, which MAY be a scope field. It MAY also contain any combination of IANA-assigned and/or Enterprise-Specific Information Element identifiers.

The format of the Options Template Record Header is shown in Figure D.

```
  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
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |         Template ID (> 255)   |         Field Count           |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |      Scope Field Count        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure D: Options Template Record Header Format

Where:

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 for an Options Template Record.

As with the Template Record Format, the only difference between the standard Options Template Record Format as defined in [RFC5101] and the MIB Object Identifier Extended Template Options Record Format is the structure of the Field Specifier (see section 4.3).

4.3 MIB OID Extended Field Specifier Format

This section specifies how the Field Specifier format in [RFC5101] is extended to allow fields to be defined using a specified MIB Object. First for a MIB Object Identifier that is a non-indexed MIB object, then for an indexed MIB object.

The Field Specifier formats are shown in Figures E to G below.

4.3.1 Standard Field Specifier Format

The Field Specifier format in figure E, along with the associated definitions, has been copied from [RFC5101], for an easier comparison with the MIB Object Identifier Extended Field Specifier Format in figures F and G.

When sending an IANA-assigned and/or Enterprise-Specific Information Element identifier, the Field Specifier Format is the same as shown below.

```
  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
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |E|  Information Element ident. |        Field Length           |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                      Enterprise Number                        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure E: Standard 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 filed MUST be present.

Information Element identifier

A numeric value that represents the type of the Information Element. Refer to [RFC5102].

Field Length

The length of the corresponding encoded Information Element, in octets. Refer to [RFC5102]. The field length may be smaller than the definition in [RFC5102] if reduced size encoding is used (see section 6.2). The value 65535 is reserved for variable length Information Element (see section 7).

Enterprise Number

IANA enterprise number [PEN] of the authority defining the Information Element identifier in this Template Record.

4.3.2 Extended Field Specifier Format for non-indexed MIB Object

When a MIB object is to be exported, a special Information Element value is used to show that the extended Field Specifier is being used, as shown in Figure F:

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|E|         MIB OID IE          |         Field Length          |
```

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Index Count = 0|MIB Obj. ID Len|    MIB Object Identifier ...  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              ... MIB Object Identifier continued              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure F: MIB Object Identifier Extended Field Specifier Format for a non-indexed MIB Object with an OID length < 255

Where:

E

Enterprise bit. In the special case of a MIB Object Identifier export, the Enterprise bit MUST always be 0, even if the exported MIB object is specified in a proprietary MIB, therefore containing the private enterprise number in its OID.

MIB OID IE

Special IPFIX Information Element, MIBObjectIdentifierMarker, that denotes that a MIB object is exported in the (Options) Template Record. When the MIB Object Identifier Information Element (MIB OID IE) is used, the MIB Object Identifier must be specified in the Field Specifier for the Collecting Process to be able to decode the Records.

Field Length

The definition is as [RFC5101].

Index Count

The number of indexes for a MIB object, and zero for a non-indexed MIB object.

MIB Object Identifier Length

The length of the MIB Object Identifier that follows. This is encoded in the same manner as the variable length encoding in [RFC5101]. If the length of the MIB Object Identifier is greater than or equal to 255 octets, the length is encoded into 3 octets before the MIB Object Name.

The first octet is 255 and the length is carried in the second and third octets (as shown in Figure H).

MIB Object Identifier

An alphanumeric-format variable name which denotes a variable name expressed as a sequence of decimal numbers or names separated by periods, as specified by the OBJECT IDENTIFIER in [RFC2578].

If the MIB Object Identifier is longer than 254 characters then the length MUST be extended:

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|E|         MIB OID IE          |         Field Length          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Index Count = 0|      255      | MIB Object Identifier Length  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  MIB Object Identifier ...                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure G: MIB Object Identifier Extended Field Specifier Format with OID length >= 255

Figure H shows the exported Template Set detailing the Template Record for exporting CPU Load (see section 3.1).

```
 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 = TBD1        |          Length = 47          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Template ID = 256      |        Field Count = 2        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|E|   IE = flowStartSeconds     |        Field Length = 4       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|   MIBObjectIdentiferMarker  |        Field Length 1         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Index Count = 0|MIB OID Len=29 |    MIB Object Identifier ...  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               = "1.3.6.1.4.1.9.9.109.1.1.1.1.7"               |
```

```
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |           ...  MIB Object Identifier continued ...           |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |           ...  MIB Object Identifier continued ...           |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |           ...  MIB Object Identifier continued ...           |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |           ...  MIB Object Identifier continued ...           |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |           ...  MIB Object Identifier continued ...           |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |           MIB Object Identifier continued     |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure H: Example of CPU Load Template Set

4.3.3   Extended Field Specifier Format for Indexed MIB Object

When an indexed MIB object is exported in IPFIX, the meaning of the exported value each index SHOULD be identified. This index (or indexes) MAY be an IPFIX Information Element or MIB Object Identifier.. Note that the IPFIX Information Element MAY be an enterprise-specific Information Element.

```
  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
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |E|         MIB OID IE          |         Field Length          |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |  Index Count  |MIB Obj. ID Len|    MIB Object Identifier ...  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |              ... MIB Object Identifier continued              |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |E| Index Information Element 1 |E| Index Information Element 2 |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                       Enterprise Number                       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 ...
```

Figure I: MIB Object Identifier Extended Field Specifier Format
with an indexed MIB Object using a IPFIX I.E. as Index

Where:

E

Enterprise bit. In the special case of a MIB Object Identifier export, the Enterprise bit MUST always be 0, even if the exported MIB object is specified in a proprietary MIB, therefore containing the private enterprise number in its OID. For any indexes identified using Information Elements the Enterprise bit can be 1, indicating that an Enterprise Number will follow the Information Element.

MIB OID IE

Special IPFIX Information Element, MIBObjectIdentifierMarker, that denotes that a MIB object is exported in the (Options) Template Record. When the MIB Object Identifier Information Element (MIB OID IE) is used, the MIB Object Identifier must be specified in the Field Specifier for the Collecting Process to be able to decode the Records.

Field Length

The definition is as [RFC5101].

Index Count

The number of indexes for a MIB object, and zero for a non-indexed MIB object.

MIB Object Identifier Length

The length of the MIB Object Identifier that follows. This is encoded in the same manner as the variable length encoding in [RFC5101]. If the length of the MIB Object Identifier is greater than or equal to 255 octets, the length is encoded into 3 octets before the MIB Object Name. The first octet is 255 and the length is carried in the second and third octets (as shown in Figure H).

MIB Object Identifier

An alphanumeric-format variable name which denotes a variable name expressed as a sequence of decimal numbers or names separated by periods, as specified by the OBJECT IDENTIFIER in [RFC2578].

A MIB Object Identifier MAY be used as an index and sent as described in Figure J. If a MIB Object Identifier with an index is used as an index then its indexes will no be identified.

```
  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
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |E|         MIB OID IE          |         Field Length          |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |  Index Count  | MIB OID Len 1 |  MIB Object Identifier 1 ...  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |             ... MIB Object Identifier 1 continued             |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |E|         MIB OID IE          | MIB OID Len 2 | MIB OID 2 ... |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |             ... MIB Object Identifier 2 continued             |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    ...
```

Figure J: MIB Object Identifier Extended Field Specifier Format with a MIB Index using a normal MIB Object Identifier as index

Where:

E

Enterprise bit. In the special case of a MIB Object Identifier export, the Enterprise bit MUST always be 0, even if the exported MIB object is specified in a proprietary MIB, therefore containing the private enterprise number in its OID.

MIB OID IE

Special IPFIX Information Element, MIBObjectIdentifierMarker, that denotes that a MIB object is exported in the (Options) Template Record. When the MIB Object Identifier Information Element (MIB OID IE) is used, the MIB Object Identifier must be specified in the Field Specifier for the Collecting Process to be able to decode the Records.

Field Length

The definition is as [RFC5101].

Index Count

The number of indexes for a MIB object, and zero for a non-indexed MIB object.

MIB Object Identifier Length 1

The length of the MIB Object Identifier being exported. This is encoded in the same manner as the variable length encoding in [RFC5101]. If the length of the MIB Object Identifier is greater than or equal to 255 octets, the length is encoded into 3 octets before the MIB Object Name. The first octet is 255 and the length is carried in the second and third octets.

MIB Object Identifier 1

An alphanumeric-format variable name of the MIB Object Identidier being exported, which denotes a variable name expressed as a sequence of decimal numbers or names separated by periods, as specified by the OBJECT IDENTIFIER in [RFC2578].

MIB Object Identifier Length 2

The length of the MIB Object Identifier being used as an index. This is encoded in the same manner as the variable length encoding in [RFC5101]. If the length of the MIB Object Identifier is greater than or equal to 255 octets, the length is encoded into 3 octets before the MIB Object Name. The first octet is 255 and the length is carried in the second and third octets.

MIB Object Identifier 1

An alphanumeric-format variable name of the MIB Object Identifier being used as an index, which denotes a variable name expressed as a sequence of decimal numbers or names separated by periods, as specified by the OBJECT IDENTIFIER in [RFC2578].

Figure K shows the exported Template Set detailing the Template for exporting a PSAMP Report with Output Queue Size (see section 3.2).

```
  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 = TBD1        |          Length               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |        Template ID = 257      |        Field Count = 5        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0|   IE = sourceIPv4Address    |        Field Length = 4       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0| IE = destinationIPv4Address |        Field Length = 4       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0|    IE = totalLengthIPv4     |        Field Length = 4       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0|    IE = egressInterface     |        Field Length = 4       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0|   MIBObjectIdentifierMark   |        Field Length 1         |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | Index Count=1 |MIB OID Len=20 |    MIB Object Identifier ...  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                     = "1.3.6.1.2.1.2.2.1.21"                  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |           ...  MIB Object Identifier continued ...            |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |           ...  MIB Object Identifier continued ...            |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |           ...  MIB Object Identifier continued ...            |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | ... MIB OID continued         |0|    IE = egressInterface     |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure K: Example of PSAMP Report with Output Queue Size

Figure L shows the exported Template Set detailing the Template for exporting a PSAMP Report with Output Queue Size but using the ifIndex MIB object as the exported index, rather than the Egress Interface Information Element.

```
  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 = TBD1        |          Length               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |        Template ID = 257      |        Field Count = 5        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0|   IE = sourceIPv4Address    |        Field Length = 4       |
```

```
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0| IE = destinationIPv4Address |        Field Length = 4       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0|    IE = totalLengthIPv4     |        Field Length = 4       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0|    IE = egressInterface     |        Field Length = 4       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0|IE=MIBObjectIdentifierMarker |        Field Length 1         |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | Index Count=1 |MIB OID Len=20 |    MIB Object Identifier ...  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                   = "1.3.6.1.2.1.2.2.1.21"                    |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |           ...  MIB Object Identifier continued ...            |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |           ...  MIB Object Identifier continued ...            |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |           ...  MIB Object Identifier continued ...            |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | ... MIB OID continued         |0|IE=MIBObjectIdentifierMarker |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |        MIB Object Identifier = "1.3.6.1.2.1.2.2.1.1"          |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |           ...  MIB Object Identifier continued ...            |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |           ...  MIB Object Identifier continued ...            |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |           ...  MIB Object Identifier continued ...            |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |      ...  MIB Object Identifier continued     |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure L: Example of PSAMP Report with Output Queue Size using ifIndex from IF-MIB as an index [RFC2578]

4.4 Template Management

Templates are managed as per [RFC5101].

The Set ID field MUST contain the value TBD1 for any Template Set that contains a MIB Object Identifier Extended Field Specifier. The Template Withdrawal Message for such a Template must also use a Set ID field containing the value TBD1.

The Set ID field MUST contain the value TBD2 for any Option Template Set that contains a MIB Object Identifier Extended

Field Specifier. The Template Withdrawal Message for such an Option Template must also use a Set ID field containing the value TBD2.

5. The Collecting Process's Side

This section describes the Collecting Process when using SCTP and PR-SCTP as the transport protocol. Any necessary changes to the Collecting Process specifically related to TCP or UDP transport protocols are specified in section 10 of RFC 5101.

The specifications in section 9 of RFC 5101 also apply to Collector's that implement this specification. In addition, the following specifications should be noted.

A Collecting Process that implements this specification MUST be able to receive Set IDs TBD1 and TBD2, as specified in this document.

A Collecting Process that implements this specification MUST have access to a MIB database in order to look up the received MIB Object IDs and find the type and name of MIB OID fields used in received templates. It should be noted that since reduced length encoding MAY be used by the Exporting Process then the Collecting Process cannot assume a received size for a field is the maximum size it should expect for that field.

If a Collecting Process receives a MIB Object ID that it cannot decode, it SHOULD log an error.

If a Collecting Process receives a MIB Object ID for an indexed MIB Object but isn't sent the appropriate number of indexes then it SHOULD log an error, but it MAY use the Template Record to decode the Data Records as the associated indexes are purely semantic information.

6. Applicability

Making available the many and varied items from the MIBs opens up a wide range of possible applications for the IPFIX protocol, some quite different from the usual flow information. Some potential enhancements for traditional applications are detailed below:

Some monitoring applications periodically export an interface id to interface name mapping using IPFIX Options Templates. This could be expanded to include the MIB Object "1.3.6.1.2.1.2.ifTable.ifEntry.ifInUcastPkts" indexed using the ingressInterface Information Element, as a index. This would give the input statistics for each interface which can be compared to the flow information to ensure the sampling rate is expected. Or, if there is no sampling, to ensure that all the expected packets are being monitored.

7. Security Considerations

For this extension to the IPFIX protocol, the same security considerations as for the IPFIX protocol apply [RFC5101].

However, the Metering Process MUST check whether or not the MIB variables can be accessed, and hence exported with IPFIX. Therefore a read or read-write community string in SNMPv1 and SNMPv2c, or a principal in SNMPv3, MUST be associated with the Metering Process.

If the management entity supports the View-based Access Control Model (VACM) for the SNMP [RFC3415], then the Metering Process MUST validate with the View-Based Access Control [RFC3415] that the MIB object can accessed before exporting his content.

If there is a view in case of SNMPv1 and SNMPv2c, the Metering Process MUST validate that the MIB object can accessed before exporting his content.
Whether the Exporter allows or not the configuration of Template that contains an unauthorized MIB object is implementation specific.

8. 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 previously reserved Set ID values of TBD1 and TBD2 are used as specified in this document. All other Set ID values are reserved for future use. Set ID values above 255 are used for Data Sets.

A new Information Element, "MIBObjectIdentifierMarker", needs to be reserved.

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

[RFC2578] McCloghrie, K. Perkins, D., Schoenwaelder, J. , Structure of Management Information Version 2 (SMIv2), RFC 2578, April 1999

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

[RFC3415] Wijnen, B., Presuhn, R., and K. McCloghrie, View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP), RFC3415, December 2002

[RFC3418] Presuhn, R., Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, Management Information Base (MIB) for the Simple Network Management Protocol (SNMP), RFC3418, December 2002

[RFC5101] B. Claise et Al, IPFIX Protocol Specification, RFC 5101, January 2008

[RFC5102] J. Quittek, S. Bryant, B. Claise, J. Meyer, Information Model for IP Flow Information Export, RFC5102, January 2008

[PEN] IANA Private Enterprise Numbers registry http://www.iana.org/assignments/enterprise-numbers.

9.2 Informative References

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

Author's Addresses

Andrew Johnson
Cisco Systems (Scotland) Ltd.
96 Commercial Quay
Commercial Street
Edinburgh, EH6 6LX, United Kingdom
Phone: +44 131 561 3641
Email: andrjohn@cisco.com

Benoit Claise
Cisco Systems
De Kleetlaan 6a b1
Diegem 1813
Belgium
Phone: +32 2 704 5622
Email: bclaise@cisco.com

Paul Aitken
Cisco Systems (Scotland) Ltd.
96 Commercial Quay
Commercial Street
Edinburgh, EH6 6LX, United Kingdom
Phone: +44 131 561 3616
Email: paitken@cisco.com

IP Flow Information Export S. Kashima
Internet-Draft A. Kobayashi
Intended status: Standards Track NTT
Expires: April 29, 2011 October 26, 2010

Information Elements for Data Link Layer Traffic Measurement
draft-kashima-ipfix-data-link-layer-monitoring-04

Abstract

This document describes Information Elements related to data link layer. They are used by the IP Flow Information Export (IPFIX) protocol for encoding measured data link layer traffic information.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction 3
  1.1. Conventions Used in This Document 3
2. Extended Ethernet Technology 3
  2.1. Wide-Area Ethernet Network Summary 3
  2.2. Data Center Network Summary 4
3. Future Traffic Measurement 4
4. New Information Elements 5
  4.1. dataLinkFrameSize 5
  4.2. dataLinkFrameSection 5
  4.3. dataLinkFrameType 6
  4.4. dataLinkFrameOffset 7
  4.5. dataLinkFrameSectionObservedOctets 7
5. Security Considerations 8
6. IANA Considerations 8
7. References 8
  7.1. Normative References 8
  7.2. Informative References 8
Appendix A. Frame Formats in Wide-Area Ethernet Network 9
Appendix B. Frame Formats in Data Center Network 14
Appendix C. Template Formats Example 16
Authors’ Addresses 16

1. Introduction

Ethernet [IEEE802.1D] and VLAN (Virtual LAN) [IEEE802.1Q] technologies used to be used only in Local Area Networks. Recently, they have been used in Wide Area Networks, e.g., L2-VPN services. Accordingly, the IEEE802.1Q standard has been enhanced to [IEEE802.1ad] and [IEEE802.1ah]. And, Ethernet in data center also has been enhanced for server virtualization and I/O consolidation.

While these renovations provide flexibility, scalability, and mobility to an existing network architecture, it increases the complexity of traffic measurement due to the existence of various Ethernet header formats. To cope with this, a more sophisticated method is required.

IPFIX/PSAMP helps to resolve these problems. However, the PSAMP Information Model [RFC5477] and the IPFIX Information Model [RFC5101] are not yet enough for Information Elements related to data link layer, e.g., Ethernet header forms. This document describes the Information Elements related to data link layers that enable a more sophisticated traffic measurement method.

1.1. 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].

2. Extended Ethernet Technology

2.1. Wide-Area Ethernet Network Summary

Provider Bridge [IEEE802.1ad] and Provider Backbone Bridge [IEEE802.1ah], which are standards for the Wide-Area Ethernet, are described below.

- In Provider Bridge [IEEE802.1ad], there are two VLAN IDs: Service VLAN Identifier (S-VID) and Customer VLAN Identifier (C-VID). S-VID is assigned to an Ethernet frame by a service provider, while C-VID is independently assigned to an Ethernet frame by a customer. Frame switching in a service provider network is based on only S-VID.
- In Provider Backbone Bridge [IEEE802.1ah], new Ethernet fields, such as Backbone VLAN Identifier (B-VID) and Backbone Service Instance Identifier (I-SID), are introduced to overcome the limitations on the VLAN identifier space on [IEEE802.1ad] and to isolate the service provider and customer identifier spaces.

Frame switching is based on a 12-bit B-VID, and customer identification is based on a 24-bit I-SID. A flexible network design has become possible because network management is separated from customer management. Other Ethernet fields that indicate quality of service (QoS) class are B-PCP, B-DEI, I-PCP, and I-DEI.

Provider Backbone Bridge enables a wide-area Ethernet service to be improved from a flat network to a hierarchical network co-existing Provider Bridge and Provider Backbone Bridge.

Frame formats used in Wide-Area Ethernet are shown in Appendix A.

2.2. Data Center Network Summary

In data center networks, Ethernet needs to be enhanced to provide the flexibility, mobility for server virtualization, and I/O consolidation. In IEEE802.1 Data Center Bridging Task Group, several Ethernet header formats are proposed to enable a simplifying networks and server managements.

The one of the enhanced methods is Bridge Port Extension [IEEE802.1Qbh], which brings a traffic exchange point to upper bridges. Bridge Port Extension introduces a Ethernet format named Multicast Replication Tag (M-TAG) in addition to existing Service VLAN Tag (S-TAG) and Customer VLAN Tag (C-TAG) to move the policy enhancement to upper bridges in data center network. On the other hand, the complexity for traffic measurement would be increased, because multiple Ethernet header formats as shown in Appendix B co-exist in the same link and in the same network.

3. Future Traffic Measurement

After the implementations of [IEEE802.1ah] and [IEEE802.1Qbh], traffic measurement methods need to absorb the complexity caused by multiple header formats.

This requirement means that it is possible to apply the IPFIX/PSAMP architecture. Therefore, we propose an Ethernet traffic measurement method using IPFIX/PSAMP, as follows.

The device (Exporter) filters and/or samples Ethernet frames using PSAMP Selector, extracts the header of the frame, and exports the header information encoded by IPFIX protocol to the Collector. The Collector sums up the number of frames and the number of octets for each Ethernet header field (e.g., B-VID, I-SID, B-PCP, I-PCP, and Replication Identifier) and for each Ethernet frame type (e.g., multicast or broadcast).

Furthermore, the device (Exporter) filters and samples traffic for each VPN using a Composite Selector of PSAMP [RFC5475]. This makes it possible to change the granularity of the traffic monitoring for each VPN.

4. New Information Elements

The following Information Elements are necessary for enabling the IPFIX/PSAMP traffic measurement for data link layer, which is not limited to Ethernet because the method can be applied to other data link protocols as well. Note that these are proposed IDs, subject to approval by IANA.

ID	Name	ID	Name
312	dataLinkFrameSize	347	dataLinkFrameType
315	dataLinkFrameSection	348	dataLinkFrameOffset
		349	dataLinkFrameSectionObservedOctets

4.1. dataLinkFrameSize

Description:

This Information Element specifies the length of the selected data link frame.

The data link layer is defined in [ISO_IEC.7498-1_1994].

Abstract Data Type: unsigned16

Data Type Semantics: quantity

ElementId: 312

Status: current

4.2. dataLinkFrameSection

Description:

This Information Element carries n octets from the data link frame of a selected frame, starting dataLinkFrameOffset octets into the frame.

When the dataLinkFrameSectionObservedOctets field corresponding to this Information Element does not exist, this Information Element MUST have a variable length and MUST NOT be padded. In this case, the size of the exported section may be constrained due to limitations in the IPFIX protocol.

When the dataLinkFrameSectionObservedOctets field corresponding to this Information Element exists, this Information Element MAY have a fixed length and MAY be padded, or MAY have a variable length.

The dataLinkFrameSectionObservedOctets expresses how much data was observed, while the remainder is padding.

Further Information Elements, i.e., dataLinkFrameType and dataLinkFrameSize are needed to specify the data link type and the size of the data link frame of this Information Element. A set of these Information Elements MAY be contained in a structured data type, as expressed in another IPFIX WG draft. Or a set of these Information Elements MAY be contained in one Flow Record as shown in Appendix C.

The data link layer is defined in [ISO_IEC.7498-1_1994].

Abstract Data Type: octetArray

ElementId: 315

Status: current

4.3. dataLinkFrameType

Description:

This Information Element specifies the type of the selected data link frame.

The following data link types are defined here.

- 0x01 ETHERNET

Further values may be assigned by IANA.

The data link layer is defined in [ISO_IEC.7498-1_1994].

Abstract Data Type: unsigned16

Data Type Semantics: identifier

ElementId: 347

Status: current

4.4. dataLinkFrameOffset

Description:

This Information Element specifies the offset of the observed dataLinkFrameSection within the data link frame. If this Information Element is omitted, it defaults to zero.

The data link layer is defined in [ISO_IEC.7498-1_1994].

Abstract Data Type: unsigned16

Data Type Semantics: quantity

ElementId: 348

Status: current

4.5. dataLinkFrameSectionObservedOctets

Description:

This Information Element specifies the observed length of the dataLinkFrameSection.

This Information Element is especially needed for NetFlow version 9 [RFC3954] because NetFlow does not support a variable-length Information Element.

The dataLinkFrameSection may be of a fixed size larger than the dataLinkFrameSectionObservedOctets. In this case, octets in the dataLinkFrameSection beyond the dataLinkFrameSectionObservedOctets MUST follow the rules for padding (ie, be composed of zero (0) valued octets).

The data link layer is defined in [ISO_IEC.7498-1_1994].

Abstract Data Type: unsigned16

Data Type Semantics: quantity

ElementId: 349

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 requests that the Information Element IDs are allocated as shown in section 4.

In addition, the dataLinkFrameType Information Element requires the creation of new IANA registries.

7. References

7.1. Normative References

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

7.2. Informative References

[IEEE802.1D]
IEEE Computer Society, "IEEE Standards for Local and Metropolitan Area Networks: Media Access Control (MAC) Bridges", IEEE Std 802.1D-2004, June 2004.

[IEEE802.1Q]
IEEE Computer Society, "IEEE Standards for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks", IEEE Std 802.1Q-2005, May 2006.

[IEEE802.1Qbh]
IEEE Computer Society, "Draft Standards for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks Amendment: Bridge Port Extension", IEEE Std P802.1Qbh/D0.4, August 2010.

[IEEE802.1ad]
IEEE Computer Society, "IEEE Standards for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks Amendment 4: Provider Bridges", IEEE Std 802.1ad-2005, May 2006.

[IEEE802.1ah]

IEEE Computer Society, "IEEE Standards for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks Amendment 7: Provider Backbone Bridges", IEEE Std 802.1ah-2008, August 2008.

[ISO_IEC.7498-1_1994]
International Organization for Standardization, "Information technology -- Open Systems Interconnection -- Basic Reference Model: The Basic Mode", ISO Standard 7498-1:1994, June 1996.

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

[RFC3954] Claise, B., "Cisco Systems NetFlow Services Export Version 9", RFC 3954, October 2004.

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

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

Appendix A. Frame Formats in Wide-Area Ethernet Network

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              C-DA                             |
+                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                              C-SA                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Length/Type          |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                                                               |
~                         Customer Data                         ~
~                                                               ~
|                                                               |
```

```
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure A-1: IEEE802.1D Frame Format in Customer Bridged Network

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                              C-DA                             |
   +                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                               |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
   |                              C-SA                             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |        C-TAG TCI=0x8100       |C-PCP|C|         C-VID         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          Length/Type          |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
   |                                                               |
   ~                         Customer Data                         ~
   ~                                                               ~
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure A-2: IEEE802.1Q Frame Format in Customer Bridged Network

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                              C-DA                             |
   +                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                               |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
   |                              C-SA                             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |        S-TAG TCI=0x88a8       |S-PCP|D|         S-VID         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          Length/Type          |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
   |                                                               |
   ~                         Customer Data                         ~
   ~                                                               ~
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure A-3: IEEE802.1ad (no C-Tag) Frame Format in Provider Bridged Network

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                              C-DA                             |
   +                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                               |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
   |                              C-SA                             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |        S-TAG TCI=0x88a8       |S-PCP|D|         S-VID         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |        C-TAG TCI=0x8100       |C-PCP|C|         C-VID         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          Length/Type          |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
   |                                                               |
   ~                         Customer Data                         ~
   ~                                                               ~
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure A-4: IEEE802.1ad (C-Tagged) Frame Format in Provider Bridged Network

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              B-DA                             |
+                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                              B-SA                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        B-TAG TCI=0x88a8       |B-PCP|D|         B-VID         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        I-TAG TCI=0x88e7       |I-PCP|D|U| Res |     I-SID     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             I-SID             |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                              C-DA                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              C-SA                             |
+                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |          Length/Type          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                         Customer Data                         ~
~                                                               ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure A-5: IEEE802.1ah (no C-Tag) Frame Format in Provider Backbone Bridged Network

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                              B-DA                             |
   +                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                               |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
   |                              B-SA                             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |        B-TAG TCI=0x88a8       |B-PCP|D|         B-VID         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |        I-TAG TCI=0x88e7       |I-PCP|D|U| Res |     I-SID     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |             I-SID             |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
   |                              C-DA                             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                              C-SA                             |
   +                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                               |        C-TAG TCI=0x8100       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |C-PCP|C|         C-VID         |          Length/Type          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   ~                         Customer Data                         ~
   ~                                                               ~
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure A-6: IEEE802.1ah (C-Tagged) Frame Format in Provider Backbone Bridged Network

Appendix B. Frame Formats in Data Center Network

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              C-DA                             |
+                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                              C-SA                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        S-TAG TCI=0x88a8       |S-PCP|D|         S-VID         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Length/Type          |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                                                               |
~                         Customer Data                         ~
~                                                               ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure B-1: IEEE802.1Qbh (S-TAG) Frame Format in Data Center Network

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              C-DA                             |
+                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                              C-SA                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        S-TAG TCI=0x88a8       |S-PCP|D|         S-VID         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        C-TAG TCI=0x8100       |C-PCP|C|         C-VID         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Length/Type          |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                                                               |
~                         Customer Data                         ~
~                                                               ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure B-2: IEEE802.1Qbh (S-TAG+C-TAG) Frame Format in Data Center Network

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              C-DA                             |
+                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                              C-SA                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        M-TAG TCI=0xXXXX       |S-PCP|D|         S-VID         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Res|    M-channel Identifier   |                Res            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Length/Type          |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                                                               |
~                         Customer Data                         ~
~                                                               ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure B-3: IEEE802.1Qbh (M-TAG) Frame Format in Data Center Network

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              C-DA                             |
+                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                              C-SA                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        M-TAG TCI=0xXXXX       |S-PCP|D|         S-VID         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Res|    M-channel Identifier   |                Res            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        C-TAG TCI=0x8100       |C-PCP|C|         C-VID         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Length/Type          |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                                                               |
~                         Customer Data                         ~
~                                                               ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure B-4: IEEE802.1Qbh (M-TAG+C-TAG) Frame Format in Data Center

Network

Appendix C. Template Formats Example

```
 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 (0x0002)        |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Template ID (0x0103)     |      Field Count (0x0006)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   ingressInterface (0x000A)   |     Field Length (0x0004)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   egressInterface (0x000E)    |     Field Length (0x0004)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|observationTimeSeconds (0x0142)|     Field Length (0x0008)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   dataLinkFrameSize (0x0138)  |     Field Length (0x0002)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| dataLinkFrameSection (0x013B) |     Field Length (0xFFFF)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   dataLinkFrameType (0x015B)  |     Field Length (0x0002)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  dataLinkFrameOffset (0x015C) |     Field Length (0x0002)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| dataLinkFrameSection (0x015D) |     Field Length (0x0002)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure C-1: Template Fortmat Example

Authors' Addresses

Shingo Kashima
NTT PF Lab.
Midori-Cho 3-9-11
Musashino-shi, Tokyo 180-8585
Japan

Phone: +81 422 59 3894
Email: kashima@nttv6.net

Atsushi Kobayashi
NTT East
Midori-Cho 3-9-11
Musashino-shi, Tokyo 180-8585
Japan

Phone: +81 422 59 3894
Email: akoba@nttv6.net

IP Flow Information Export S. Kashima
Internet-Draft NTT
Intended status: Standards Track October 11, 2010
Expires: April 14, 2011

Information Elements for Short Timer
draft-kashima-ipfix-short-timer-00

Abstract

This document describes Information Elements related to short timer. They are used by the IP Flow Information Export (IPFIX) protocol for encoding timer paramerters required for traffic measurment of volume change in a short time.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction 3
2. Terminology 3
2.1. IPFIX Documents Overview 3
2.2. PSAMP Documents Overview 4
3. Existing Information Elements 4
4. New Information Elements 6
4.1. flowActiveTimeoutMilliseconds 6
4.2. flowIdleTimeoutMilliseconds 6
5. Security Considerations 6
6. IANA Considerations 7
7. References 7
7.1. Normative References 7
7.2. Informative References 7
Author’s Address 8

1. Introduction

The IPFIX Information Model [RFC5102] defines an extensible list of Information Elements which may be transmitted by the IPFIX protocol [RFC5102].

This document lists a series of new Information Elements to update the IPFIX Information Model, and acts as the persistent publication medium requested in the IANA considerations section of the IPFIX Information Model [RFC5102] ("The specification of new IPFIX Information Elements MUST use the template specified in section 2.1 and MUST be published using a well established and persistent publication medium").

2. Terminology

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

2.1. IPFIX Documents Overview

The IPFIX Protocol [RFC5101] provides network administrators with access to IP flow information.

The architecture for the export of measured IP flow information out of an IPFIX exporting process to a collecting process is defined in 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].

Finally 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.

2.2. PSAMP Documents Overview

The document "A Framework for Packet Selection and Reporting" [RFC5474], describes the PSAMP framework for network elements to selectsubsets of packets by statistical and other methods, and to export a stream of reports on the selected packets to a collector.

The set of packet selection techniques (sampling, filtering, and hashing) supported by PSAMP are described in "Sampling and Filtering Techniques for IP Packet Selection" [RFC5475].

The PSAMP protocol [RFC5476] specifies the export of packet information from a PSAMP Exporting Process to a PSAMP Collecting Process. Like IPFIX, PSAMP has a formal description of its information elements, their name, type and additional semantic information. The PSAMP information model is defined in [RFC5477].

Finally [I-D.ietf-ipfix-psamp-mib] describes the PSAMP Management Information Base.

3. Existing Information Elements

The following are existing Information Elements related to time stamp or time duration. Becasue one Application of IPFIX is QoS (Quality of service) monitoring, they support units smaller than seconds. For example, we can use observationTimeMilliseconds for delay measurements.

ID	Name	Units
21	flowEndSysUpTime	milliseconds
22	flowStartSysUpTime	milliseconds
36	flowActiveTimeout	seconds
37	flowIdleTimeout	seconds
150	flowStartSeconds	seconds
151	flowEndSeconds	seconds
152	flowStartMilliseconds	milliseconds
153	flowEndMilliseconds	milliseconds
154	flowStartMicroseconds	microseconds
155	flowEndMicroseconds	microseconds
156	flowStartNanoseconds	nanoseconds
157	flowEndNanoseconds	nanoseconds
158	flowStartDeltaMicroseconds	microseconds
159	flowEndDeltaMicroseconds	microseconds
160	systemInitTimeMilliseconds	milliseconds
161	flowDurationMilliseconds	milliseconds
162	flowDurationMicroseconds	microseconds
258	collectionTimeMilliseconds	milliseconds
260	maxExportSeconds	seconds
261	maxFlowEndSeconds	seconds
264	minExportSeconds	seconds
265	minFlowStartSeconds	seconds
268	maxFlowEndMicroseconds	microseconds
269	maxFlowEndMilliseconds	milliseconds
270	maxFlowEndNanoseconds	nanoseconds
271	minFlowStartMicroseconds	microseconds
272	minFlowStartMilliseconds	milliseconds
273	minFlowStartNanoseconds	nanoseconds
305	samplingTimeInterval	microseconds
306	samplingTimeSpace	microseconds
322	observationTimeSeconds	seconds
323	observationTimeMilliseconds	milliseconds
324	observationTimeMicroseconds	microseconds
325	observationTimeNanoseconds	nanoseconds

According to a diversification of multimedia application and an aggregation of server in data center, we are facing to measure bursty traffic that causes packets loss and delay jitter. In order to measure bursty traffic with IPFIX/PSAMP, timers shorter than one second are required.

4. New Information Elements

The following Information Elements are necessary for enabling the IPFIX/PSAMP traffic measurement of volume change in a short time.

ID	Name	Units
TBD1	flowActiveTimeoutMilliseconds	milliseconds
TBD2	flowIdleTimeoutMilliseconds	milliseconds

4.1. flowActiveTimeoutMilliseconds

Description:

The number of milliseconds after which an active Flow is timed out anyway, even if there is still a continuous flow of packets.

Abstract Data Type: unsigned16

ElementId: TDB1

Status: current

Units: milliseconds

4.2. flowIdleTimeoutMilliseconds

Description:

A Flow is considered to be timed out if no packets belonging to the Flow have been observed for the number of seconds specified by this field.

Abstract Data Type: unsigned16

ElementId: TDB1

Status: current

Units: milliseconds

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 ElementId assignment to be made by IANA.

7. References

7.1. Normative References

7.2. Informative References

[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-01 (work in progress), July 2010.

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

[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.

[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.

[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.

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

Author's Address

Shingo Kashima
NTT PF Lab.
Midori-Cho 3-9-11
Musashino-shi, Tokyo 180-8585
Japan

Phone: +81 422 59 3894
Email: kashima@nttv6.net

IPFIX Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 28, 2011

B. Trammell
E. Boschi
ETH Zurich
A. Wagner
Consecom AG

October 25, 2010

Exporting Aggregated Flow Data using the IP Flow Information Export (IPFIX) Protocol

draft-trammell-ipfix-a9n-01.txt

Abstract

This document describes the export of aggregated Flow information using IPFIX. An Aggregated Flow is essentially an IPFIX Flow representing packets from zero or more original Flows, within an externally imposed time interval. The document describes Aggregated Flow export within the framework of IPFIX Mediators and defines an interoperable, implementation-independent method for Aggregated Flow export.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction 3
1.1. Rationale and Scope 4
2. Terminology 5
3. Use Cases for IPFIX Aggregation 6
4. Aggregation of IP Flows 7
4.1. A note on temporal and spatial aggregation 7
4.2. A general operational model for IP Flow aggregation . . . 8
4.3. Interval Distribution 9
4.4. Key Aggregation 10
4.5. Aggregating and Distributing Counters 12
4.6. Counting Original Flows 14
4.7. Counting Distinct Key Values 14
4.8. Exact versus Approximate Counting during Aggregation . . . 15
4.9. Time Composition 15
5. Aggregation in the IPFIX Architecture 15
6. Export of Aggregated IP Flows using IPFIX 17
6.1. Time Interval Export 17
6.2. Flow Count Export 18
6.2.1. originalFlowsPresent Information Element 18
6.2.2. originalFlowsInitiated InformationElement 18
6.2.3. originalFlowsCompleted InformationElement 18
6.2.4. originalFlows InformationElement 19
6.3. Aggregate Counter Distibution Export 19
6.3.1. Aggregate Counter Distribution Options Template . . . 19
6.3.2. valueDistributionMethod Information Element 20
7. Examples 21
8. Security Considerations 21
9. IANA Considerations 21
10. Acknowledgments 22
11. References 22
11.1. Normative References 22
11.2. Informative References 22
Authors' Addresses 23

1. Introduction

The aggregation 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 [RFC5101]. 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 [I-D.ietf-ipfix-mediators-framework].

The Mediator framework offered an initial but inexhaustive treatment of the topic of aggregation. This document expands on the definitions presented there, providing an implementation-neutral, interoperable specification of an Intermediate Aggregation Process which can operate within the Mediator framework or independent thereof.

Aggregation is part of a wide variety of applications, including traffic matrix calculation, generation of time series data for visualizations or anomaly detection, and data reduction. Depending on the keys used for aggregation, it may have an anonymising affect on the data. Aggregation can take place at one of any number of locations within a measurement infrastructure. Exporters may export aggregated Flow information simply as normal flow information, by performing aggregation after metering but before export. IPFIX Mediators are particularly well suited to performing aggregation, as they can collect information from multiple original exporters at geographically and topologically distinct observation points.

Aggregation as defined and described in this document covers a superset of the applications defined in [RFC5982], including 5.1 "Adjusting Flow Granularity (herein referred to as Key Aggregation), 5.4 "Time Composition" (herein referred to as Interval Combination), and 5.5 "Spatial Composition".

Note that an Intermediate Aggregation process may be applied to data collected from multiple Observation Points, as aggregation is natural to apply for data reduction when concentrating measurement data. This document specifically does not address the architectural and protocol issues that arise when combining IPFIX data from multiple Observation Points and exporting from a single Mediator, as these issues are general to Mediation in general. These are treated in detail in the Mediator Protocol [I-D.claise-ipfix-mediation-protocol] document.

Since aggregated flows as defined in the following section are essentially Flows, IPFIX can be used to export [RFC5101] and store [RFC5655] aggregated data "as-is"; there are no changes necessary to the protocol. However, this document further provides a common basis for the application of IPFIX to the handling of aggregated data, through a detailed terminology, model of aggregation operations, methods for original Flow counting and counter distribution across time intervals, and an aggregation metadata representation based upon IPFIX Options.

1.1. Rationale and Scope

This specification of Aggregated Flow export has interoperability and implementation-independence as its two key goals. First, export of Aggregated Flows using the techniques described in this document will result in Flow data which can be collected by Collecting Processes and read by File Readers which do not provide any special support for Aggregated Flow export. An Aggregated Flow is simply a Flow with some additional conditions as to how it is derived.

Second, in Section 5, we specify aggregation in an implementation-independent way. While we must describe the aggregation process in terms of operations due to the interdependencies among them, these operations like the stages in the IPFIX Architecture [RFC5470] are meant to be descriptive as opposed to proscriptive. We specify the flow aggregation process as an intermediate process within the IPFIX Mediator framework [I-D.ietf-ipfix-mediators-framework], and specify a variety of different architectural arrangements for flow aggregation. When exporting aggregation-relevant metadata, we seek to define properties of the set of exported Aggregated Flows, as opposed to the properties of the specific algorithms used to aggregate these Flows. Specifically out of scope for this effort are any definition of a language for defining aggregation operations, or the configuration parameters of Aggregation Processes, as these are necessarily implementation dependent.

From the definition of presented below in Section 2, an Aggregated Flow is a Flow as in [RFC5101], with additional conditions as to the packets making up the Flow. Practically speaking, Aggregated Flows are derived from original Flows, as opposed to a raw packet stream. Key to this definition of Aggregated Flow is how timing affects the process of aggregation, as for the most part flow aggregation takes place within some set of time intervals, which are usually regular and externally imposed, or derived from the flows themselves. Aggregation operations concerning keys, which are often called "spatial aggregation" in the literature, will necessarily impact and be impacted by these time intervals; aggregation operations concerning these time intervals are often called "temporal aggregation" in the literature. Prior definitions of aggregation attempt to treat temporal and spatial aggregation separately; this document recognizes that this is not possible due to the interdependencies between flows and their time intervals, and defines these operations as interdependent.

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 [RFC2119].

In addition, this document defines the following terms

Aggregated Flow: A Flow, as defined by [RFC5101], derived from a set of zero or more original Flows within a defined time interval. The two primary differences between a Flow and an Aggregated Flow are (1) that the time interval of a Flow is generally derived from information about the timing of the packets comprising the Flow, while the time interval of an Aggregated Flow are generally externally imposed; and (2) that an Aggregated Flow may represent zero packets (i.e., an assertion that no packets were seen for a given Flow Key in a given time interval).

(Intermediate) Aggregation Function: A mapping from a set of zero or more original Flows into a set of Aggregated Flows accross one or more time intervals.

(Intermediate) Aggregation Process: An Intermediate Process, as in [I-D.ietf-ipfix-mediators-framework], hosting an Intermediate Aggregation Function. Note that this definition, together with that given above, updates the definition given in [I-D.ietf-ipfix-mediators-framework] to account for the more precise definition of Aggregated Flow given herein. An Aggregation Process need not be intermediate; that is, while Aggregation Processes will often be deployed within a Mediator, this is not necessarily the case.

Aggregation Interval: A time interval imposed upon an Aggregated Flow. Aggregation Functions 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 flows being aggregated.

original Flow: A Flow given as input to an Aggregation Function 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.

3. Use Cases for IPFIX Aggregation

Aggregation, as a common data analysis method, has many applications. When used with a regular Aggregation Interval, it generates time series data from a collection of flows with discrete intervals. Time series data is itself useful for a wide variety of analysis tasks, such as generating parameters for network anomaly detection systems, or driving visualizations of volume per time for traffic with specific characteristics. Traffic matrix calculation from flow data is inherently an aggregation action, by aggregating the flow key down to interface, address prefix, or autonomous system.

Irregular or data-dependent Aggregation Intervals and Key Aggregation operations can be also be used to provide adaptive aggregation of network flow data, providing a lower-resolution view (i.e. more aggregation) on data deemed "less interesting" to a given application, while allowing higher resolution (i.e. less or no aggregation) for data of interest. For example, in a 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.

Note that an aggregation operation which removes potentially sensitive information as identified in [I-D.ietf-ipfix-anon] 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 Section 4 of [I-D.ietf-ipfix-anon] are addressed.

4. Aggregation of IP Flows

As stated in Section 2, an Aggregated Flow is simply an IPFIX Flow generated from original Flows by an Aggregation Function. Here, we discuss temporal and spatial aspects of aggregation, present a general model for aggregation, and elaborate and provide examples of specific aggregation operations that may be performed by the Aggregation Process; we use this to define the export of Aggregated Flows in Section 6

4.1. A note on temporal and spatial aggregation

In general, aggregation of data records bearing time information can take place in time (by grouping the original records by time) or in space (by grouping the original records by some other dimension; in the case of IP Flows, this would generally be a flow key.

Temporal aggregation is treated in [I-D.ietf-ipfix-mediators-framework] in section 5.3.2.3, as "[m]erging a set of Data Records within a certain time period into one Flow Record by summing up the counters where appropriate," as well as in the definition of "temporal composition, wherein "multiple consecutive Flow Records with identical Flow Key values are merged into a single Flow Record of longer Flow duration if they arrive within a certain time interval."

Spatial aggregation is treated in [I-D.ietf-ipfix-mediators-framework] in section 5.3.2.3, as "spatial composition", wherein "Data Records sharing common properties are merged into one Flow Record within a certain time period." Even this definition hints at the problem in attempting to treat temporal and spatial aggregation of IP flow data orthogonally.

The issue arises because an IP Flow, as defined in [RFC5101], has three types of properties: flow keys, which "define" the properties common to all packets in the Flow; flow values or non-key fields, which describe the Flow itself; and the time interval of the Flow. The keys and time interval serve to uniquely identify the Flow. When spatially aggregating Flows, these Flows bring their time intervals along with them. The time intervals of the spatially aggregated Flows must either be combined through union, or externally imposed by splitting the original Flow across one or more

To address this subtle interdependency, it is more useful to view an Aggregation Function in terms of the temporal operations of the function, called "interval distribution" herein; and the spatial operations of the function, called "key aggregation" herein; this follows in the general model presented in the following subsection.

4.2. A general operational model for IP Flow aggregation

An Intermediate Aggregation Process consumes original Flows and exports Aggregated Flows, as defined in Section 2. While this document does not define an implementation of an Intermediate Aggregation Process further than this, or the Aggregation Functions that it applies, it can be helpful to partially decompose this function into a set of common operations, in order to more fully examine the effects these operations have.

Aggregation is composed of three general types of operations on original Flows: those that externally impose a time interval, called here the Aggregation Interval; those that derive a new Flow Key for the Aggregated Flows from the original Flow information; and those that aggregate and distribute the resulting non-Flow Key fields accordingly. Most aggregation functions will perform each of these types of operations.

Interval distribution is the external imposition of a time interval onto an original Flow. Note that this may lead to an original Flow contributing to multiple aggregated Flows, if the original Flow's time interval crosses at least one boundary between Aggregation Intervals. Interval Distribution is described in more detail in Section 4.3.

Key aggregation, the derivation of Flow Keys for Aggregated Flows from original Flow information, is made up of two operations: reduction and replacement. Reduction removes Information Elements from the original Flow Key, or otherwise constrains the space of values in the Flow Key (e.g., by replacing IP addresses with /24 CIDR blocks). In replacement, Information Elements derived from fields in the original Flow itself may be added to the Flow Key. Both of these modifications may result in multiple original Flows contributing to the same Aggregated Flow. Key Aggregation is described in more detail in Section 4.4.

Interval distribution and key aggregation together may generate multiple intermediate aggregated Flows covering the same time interval with the same Flow Key; these intermediate Flows must therefore be combined into Aggregated Flows. Non-key values are first distributed among the Aggregated Flows to which an original Flow contributes according to some distribution algorithm (see Section 4.5), and 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: in general, counters are added, averages are averaged, flags are unioned, and so on. Key aggregation may also introduce new non-key fields, e.g. per-flow average counters, or distinct counters for key fields reduced out of the Aggregated Flow.

As a result of this final combination and distribution,an Aggregation Function produces at most one Aggregated Flow resulting from a set of original Flows for a given Aggregated Flow Key and Aggregation Interval.

This general model is illustrated in the figure below. Note that within an implementation, these steps may occur in any order, and indeed be combined together in any way.

```
                        +-----------------------+
                     +->| Interval distribution |-+
                     |  +-----------------------+ |
                     |            ^  (partially   |
                     |            |   aggregated  |
                     |            V     flows)    |
                     |  +-----------------+       |
   original Flows -+->| Key aggregation |----+  |
                        +-----------------+    |  |
                                               V  V
                                +--------------------+
                                |  Combination of    |
                                | contributing Flows |
                                +--------------------+
                                          |
                                          V
                               +----------------------+
                               | Counter Distribution |
                               +----------------------+
                                          |
                                          V
                                   Aggregated Flows
```

Figure 1: Conceptual model of aggregation operations

4.3. Interval Distribution

Interval Distribution imposes a time interval on the resulting Aggregated Flows. The selection of an interval is a matter for the specific aggregation application. Intervals may be derived from the flows themselves (e.g, an interval may be selected to cover the entire interval containing the set of all flows sharing a given Key) 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. and has tradeoffs. Shorter intervals allow higher resolution aggregated data and, in streaming applications, faster reaction time. Longer intervals 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.

```
|                |                |                |
| |<--flow A-->| |                |                |
|       |<--flow B-->|            |                |
|         |<-------------flow C-------------->|    |
|                |                |                |
|   interval 0   |   interval 1   |   interval 2   |
```

Figure 2: Illustration of interval distribution

In Figure 2, 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 flows, and must have its counters distributed according to some policy as in Section 4.5.

4.4. Key Aggregation

Key Aggregation generates a new Flow Key for the Aggregated Flows from the original Flow Keys, 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, an Aggregated Flow Key may be derived through replacement, e.g. by removing one or more fields from the original

Flow and replacing them with a fields derived from the removed fields. Replacement may refer to external information (e.g., IP to AS number mappings). Replacement need not replace only key fields; for example, an application aggregating byte counts per flow size in packets would promote the packet count to a Flow Key field.

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 4.6 and Section 4.7, 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. Key Aggregation may aggregate original Flows with different sets of Flow Key fields; only the Flow Keys of the resulting Aggregated Flows of any given Key Aggregation operation need contain the same set of fields.

```
Original Flow Key
+---------+---------+----------+----------+-------+-----+
| 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 Key (by source address and destination class-C)
```

Figure 3: Illustration of key aggregation by reduction

Figure 3 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 low 8 bits.

```
Original Flow Key
+---------+---------+----------+----------+-------+-----+
| 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 Key (by source and dest ASN)
```

Figure 4: Illustration of key aggregation by reduction and replacement

Figure 4 illustrates an example reduction and replacement operation, aggregation by source and destination ASN without ASN information available in the original Flow. Here, the port, protocol, and type-of-service information is removed from the flow key, while the source and destination addresses are run though an IP address to ASN lookup table, and the Aggregated Flow key is made up of the resulting source and destination ASNs.

4.5. Aggregating and Distributing Counters

In general, counters in Aggregated Flows are treated the same as in any Flow. Each counter is independently is 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.

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 Key), and this number is added to each corresponding counter in each Aggregated Flow.

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.

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, bulk transfer flows 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 6.3. 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 (e.g., inaccuracy introduced by the use of floating-point numbers is tolerable). This property allows data to be aggregated and re-aggregated without any 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.

4.6. 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 6.2 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, 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 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 2. 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 0). 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.

4.7. Counting Distinct Key Values

One common case in aggregation is counting distinct values that were reduced away during key aggregation. For example, consider an application counting destinations contacted per host, a common case in host characterization or anomaly detection. Here, the Aggregation Process needs a way to export this distinct key count information.

For such applications, a distinctCountOf(key name) Information Element should be registered with IANA to represent these cases. [EDITOR'S NOTE: There is an open question as to the best way to do this: either through the registration of Information Elements for common cases in this draft, the registration of Information Elements on demand, or the definition of a new Information Element space for distinct counts bound to a PEN, as in [RFC5103].]

4.8. 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.

4.9. 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 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. Aggregation in the IPFIX Architecture

The techniques described in this document can be applied to IPFIX data at three stages within the collection infrastructure: on initial export, within a mediator, or after collection, as shown in Figure 5.

[EDITOR'S NOTE: determine where this lives: in the introduction or down here? Note explicitly that an IAP may live outside a mediator. Check both these figures for parallels to mediator framework.]

```
   +==========================================+
   | Exporting Process                        |
   +==========================================+
     |                                      |
     |             (Aggregated Flow Export) |
     V                                      |
   +=============================+          |
   | Mediator                    |          |
   +=============================+          |
     |                                      |
     | (Aggregating Mediator)               |
     V                                      V
   +==========================================+
   | Collecting Process                       |
   +==========================================+
           |
           | (Aggregation for Storage)
           V
   +--------------------+
   | IPFIX File Storage |
   +--------------------+
```

Figure 5: Potential Aggregation Locations

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

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, aggregation can be applied as well to aggregating original Flows from multiple sources 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.claise-ipfix-mediation-protocol] for details.

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.

The data flows into and out of an Intermediate Aggregation Process are showin in Figure 6.

```
packets --+                   +- IPFIX Messages -+
          |                   |                  |
          V                   V                  V
+==================+ +=====================+ +==============+
| Metering Process | | Collecting Process  | | File Reader  |
|                  | +=====================+ +==============+
|                  |            |  original Flows  |
|                  |            V                  V
+ - - - - - - - - -+======================================+
|           Intermediate Aggregation Process (IAP)        |
+=========================================================+
          | Aggregated                  Aggregated |
          | Flows                            Flows |
          V                                        V
+===================+                       +==============+
| Exporting Process |                       | File Writer  |
+===================+                       +==============+
          |                                        |
          +------------> IPFIX Messages <----------+
```

Figure 6: Data flows through the aggregation process

6. 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.

6.1. 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 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.

6.2. Flow Count Export

The following four Information Elements are defined to count original Flows as discussed in Section 4.6.

6.2.1. originalFlowsPresent Information Element

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: Proposed

6.2.2. originalFlowsInitiated InformationElement

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

Abstract Data Type: unsigned64

ElementId: TBD2

Status: Proposed

6.2.3. originalFlowsCompleted InformationElement

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

Abstract Data Type: unsigned64

ElementId: TBD3

Status: Proposed

6.2.4. originalFlows InformationElement

Description: The conservative count of original Flows contributing to this Aggregated Flow; may be distributed via any of the methods described in Section 4.5.

Abstract Data Type: float64

ElementId: TBD4

Status: Proposed

6.3. Aggregate Counter Distibution Export

When exporting counters distributed among Aggregated Flows, as described in Section 4.5, the Exporting Process MAY export an Aggregate Counter Distribution 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 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.

6.3.1. 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. 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.

6.3.2. 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 Template. The method is deemed to apply to all the non-key Information Elements in the referenced Template 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. The distribution methods are taken from Section 4.5 and encoded as follows:

Value	Description
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: TBD5

Status: Proposed

7. Examples

[TODO]

8. Security Considerations

[TODO]

9. IANA Considerations

[TODO: add all IEs defined in Section 6.]

10. Acknowledgments

Many thanks to Benoit Claise for his thorough review of this work. This work is materially supported by the European Union Seventh Framework Programme under grant agreement 257315 (DEMONS).

11. References

11.1. 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.

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

11.2. Informative References

[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.

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

[RFC5153] Boschi, E., Mark, L., Quittek, J., Stiemerling, M., and P. Aitken, "IP Flow Information Export (IPFIX) Implementation Guidelines", RFC 5153, April 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.

[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.

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

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

[I-D.ietf-ipfix-anon]
Boschi, E. and B. Trammell, "IP Flow Anonymisation Support", draft-ietf-ipfix-anon-05 (work in progress), October 2010.

[I-D.ietf-ipfix-mediators-framework]
Kobayashi, A., Claise, B., Muenz, G., and K. Ishibashi, "IPFIX Mediation: Framework", draft-ietf-ipfix-mediators-framework-08 (work in progress), August 2010.

[I-D.claise-ipfix-mediation-protocol]
Claise, B., Kobayashi, A., and B. Trammell, "Specification of the Protocol for IPFIX Mediations", draft-claise-ipfix-mediation-protocol-01 (work in progress), March 2010.

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

Elisa Boschi
Swiss Federal Institute of Technology Zurich
Gloriastrasse 35
8092 Zurich
Switzerland

Email: boschie@tik.ee.ethz.ch

Arno Wagner
Consecom AG
Bellariastrasse 12
8002 Zurich
Switzerland

Email: arno@wagner.name

IPFIX Working Group   B. Trammell
Internet-Draft   ETH Zurich
Intended status: BCP   B. Claise
Expires: April 4, 2011   Cisco Systems
                                                         October 1, 2010

Guidelines for Authors and Reviewers of IPFIX Information Elements
draft-trammell-ipfix-ie-doctors-00.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

Copyright Notice

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 6
4.2. Information Element data types 7
4.3. Ancillary Information Element properties 7
4.4. Internal structure in Information Elements 8
4.5. Enumerated Values and Subregistries 9
4.6. Reversibility as per RFC 5103 9
5. The Information Element Lifecycle: Revision and Deprecation . 10
6. When not to define new Information Elements 12
6.1. Maximizing reuse of existing Information Elements 12
6.2. Applying enterprise-specific Information Elements 13
7. Applying IPFIX to non-Flow Applications 14
8. Defining Recommended Templates 14
9. A Textual Format for Specifying Information Elements and Templates 15
9.1. Information Element Specifiers 16
9.2. Specifying Templates 18
9.3. Specifying IPFIX Structured Data 18
10. Security Considerations 19
11. IANA Considerations 20
12. Acknowledgements 20
13. Open Issues 20
14. References 20
14.1. Normative References 20
14.2. Informative References 21
Authors' Addresses 22

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 is largely defined through the definition of new Information Elements beyond those defined in the IPFIX Information Model [RFC5102] or already added to 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 a set of guidelines and best practices to be used in deciding which Information Elements are necessary for a given 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 specifies additional practices beyond those appearing in the IANA Considerations sections of existing IPFIX documents, especially the Information Model [RFC5102]. The practices outlined in this document are intended to guide experts when making changes to the IANA registry under Expert Review as defined in [RFC5226].

1.2. Overview of relevant IPFIX documents

[RFC5101] 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.

[RFC5102] defines the initial IPFIX Information Model, as well as procedures for extending the Information Model. It states that new Information Elements may be added to the Information Model on Expert Review basis, and delegates the appointment of experts to an IESG Area Director. 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 implicit restrictions on the use of data types and semantics; these restrictions MUST be observed in the definition of new Information Elements, as in Section 4.3.

2. Terminology

Capitalized terms used in this document that are defined in the Terminology section of [RFC5101] 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 [RFC5101].

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:

- 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.

- 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.

- 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.

- 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 must 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 8. Section 9 defines a compact textual Information Element notation to be used in describing these suggested Templates and/or the use of IPFIX Structured Data [I-D.ietf-ipfix-structured-data] 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:

- The Information Element MUST be sufficiently unique within the registry. A proposed Information Elements which is a substantial duplicate of an exiting Information Element is to be represented using the existing Element.

- 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 4.4.

- 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 6.2 of [RFC5101].

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 [RFC5102] as appropriate.

4.1. Information Element naming

Information Element Names should be defined in accordance with section 2.3 of [RFC5102]; the most important naming conventions are repeated here for convenience.

- Names of Information Elements should be descriptive.

- Names of Information Elements MUST be unique within the IPFIX information model.

- Names of Information Elements start with non-capitalized letters.

- Composed names 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'. 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.

Because IPFIX provides reduced-length encoding for Information Elements, unless an integral Information Element is derived from a fixed-width field in a measured protocol (e.g., tcpSequenceNumber, which is an unsigned32), it should be defined with the maximum possible width, generally signed64 or unsigned64. Applications can then choose to use reduced-size encoding as defined in Section 6.2 of [RFC5101] in cases where fewer than 2^64 values are necessary.

Information Elements representing time values should be exported 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.

4.3. Ancillary Information Element properties

Information Elements with numeric types and special semantics SHOULD define these semantics with one of the values in the Information Element Semantics registry, as described in Section 3.2 of [RFC5102], subject to the restrictions given in Section 3.10 of [RFC5610]; essentially, 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.

4.4. Internal structure in Information Elements

Unless defining an Information Element which is a direct copy of a bitfield or other structured entity (e.g., the tcpControlBits Information Element for the Flags byte from the TCP header) in a measured protocol, the definition of Information Elements with internal structure with the structure defined in the Description field is discouraged. In this case, the field SHOULD be decomposed into multiple primitive Information Elements to be used in parallel. For more complicated semantics, where the structure may not have use the IPFIX Structured Data [I-D.ietf-ipfix-structured-data] 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". Indeed, 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. A specific example of this within the IANA registry is 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 [I-D.ietf-ipfix-structured-data] where supported, as a basicList of MPLS label entries.

Note that a Template may contain multiple instances of the same Information Element; in this case, the each of the Information Elements in the Template are semantically indistinguishable, and appear in their "natural" order, where natural order is defined according to application; PSAMP uses this for exporting selectors. Multiple IEs used in this way are preferable to IEs with internal structure, but only when there is some natural order, and no semantic interdependence among the elements.

4.5. 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].

4.6. Reversibility as per RFC 5103

[TODO: fix this para][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. Section 6.1 of [RFC5103] states that CPs should use the set of criteria therein to determine reversibility. Since almost all Information Elements are reversible, these criteria are expressed as to determine the exceptions, i.e. which Information Elements are NOT reversible.

To ease the determination of reversibility, future Information Elements which are NOT reversible SHOULD note this fact in the description at the time of definition.

5. The Information Element Lifecycle: Revision and Deprecation

The Information Element status field in the Information Element Registry is defined in [RFC5102] to allow Information Elements to be 'deprecated' or 'obsolete'. No Information Elements are as of this writing deprecated, and but provides no further explanation of these statuses, [RFC5102] 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:

- it involves the correction of an error which is obviously only editorial; or
- it corrects an ambiguity in the Information Element's definition, which itself leads to non-interoperability (e.g., a prior change to ipv6ExtensionHeaders); or
- 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
- it defines a previously undefined or reserved enumerated value, or one or more previously reserved bits in an Information Element with flag semantics; or
- it expands the set of permissible values in the Information Element's range; or
- it harmonizes with an external reference which was itself corrected.

A non-interoperable Information Element change may also be made if it can be reasonably assumed in the eyes of the appointed experts that no unchanged implementation of the Information Element exists; this can be held to happen if a non-interoperable change to an Information Element defined shortly before is proposed to the IPFIX mailing list by the original proposer of the Information Element, and no objection is raised within a reasonable amount of time, to be defined by the expert reviewers.

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. Changes that are not permissible MUST 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 appointed experts and a responsible Operations Area Director for review; if there is no objection to the change from any appointed expert, IANA makes the change in the Information Element Registry according to its internal procedures. 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.

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 appointed 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 Information Elements MUST NOT be reused. Names of obsolete Information Elements MAY be reused, but this is NOT RECOMMENDED, as it may cause confusion among users.

6. When not to define new Information Elements

Also important in defining new applications is avoiding redundancy and clutter in the Information Element registry. Here we 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.

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.

Specifically, existing timestamp and time range Information Elements should be reused for any situation requiring simple timestamping 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 Messages. Timestamps based on information which must be exported in a separate 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.

The best practice in Information Element creation is a conservative one: don't create a new Information Element unless you really need it.

6.2. Applying enterprise-specific Information Elements

IPFIX provides a mechanism for defining enterprise-specific Infomation Elements, as in Section 3.2 of [RFC5101]. 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.

7. 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 handled by IPFIX Options. Here, the Information Elements within an Options Template are split into Scope IEs which define the key, and non-scope IEs which define the values associated with that key. Unlike Flows, Options are not necessarily scoped in time; an Option is generally held to be in effect until a new set of values for a specific set of keys is exported. While Options are often used by IPFIX to export metadata about the collection infrastructure, they are applicable to any association information.

An IPFIX application can mix Flow Records and Options 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.

8. 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 Templates or Options Templates necessary for supporting the application, as well as examples of records exported using these Templates. In defining these Templates, 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 Templates 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 [RFC5101] for exporting export statistics; in the file format [RFC5655] for exporting file metadata; and in Mediator intermediate process definitions such as [I-D.ietf-ipfix-anon] for intermediate process metadata. The discussion in these examples is a good model for recommended template definitions.

However, the bitmap diagrams of these Templates are illustrative 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 have defined RECOMMENDED textual format for specifying Information Elements and Templates in Internet-Drafts in Section 9.

9. A Textual Format for Specifying Information Elements and Templates

The extension of IPFIX will generate a fair amount of documentation and discussion covering the definition of new Information Elements. 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.

9.1. 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:

- () parentheses for Information Element numbers,
- <> angles for Information Element data types, and
- [] square brackets for Information Element sizes.
- {} 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 9.2

The symbol + is reserved for Information Element nesting within structured data elements; these are described in and Section 9.3, 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:

- octetDeltaCount

- 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)

9.2. 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}
```

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 of [RFC5101] in IESpec syntax is shown below:

```
templateId(145)<unsigned16>[2]{scope}
flowKeyIndicator(173)<unsigned64>[8]
```

9.3. Specifying IPFIX Structured Data

IESpecs can also be used to illustrate the structure of the information exported using the IPFIX Structured Data extension [I-D.ietf-ipfix-structured-data]. 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 parially qualified IESpecs as follows:

```
basicList{oneOrMoreOf}
+sourceIPv4Address(8)[4]
```

And an example subTemplateList itself containing a basicList is shown below:

```
subTemplateList{allOf}
+basicList{oneOrMoreOf}
++sourceIPv4Address(8)[4]
+destinationIPv4Address(12)[4]
```

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 [I-D.ietf-ipfix-structured-data]; 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 [I-D.ietf-ipfix-structured-data].

10. 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 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 MAY also be stressed in a separate draft. 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."

11. IANA Considerations

[TODO - collect IANA considerations from the document once we have them.]

12. Acknowledgements

[TODO]

13. Open Issues

- o add examples everywhere (including sipclf)
- o explain the range 0-127.
- o explain that existing draft should use temporary IE identifier such as XXX, YYY, and ZZZ both in the text and in the examples, and a note to IANA: "to be replaced by IANA when the IE identifier is assigned"
- o TBD (in WG): Do we want the IE-Doctors to be a formal directorate under the OPS area? What can we take from the experience of PMOL?

14. References

14.1. 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.

[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.

[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.

14.2. Informative References

[RFC2804] IAB and IESG, "IETF Policy on Wiretapping", RFC 2804, May 2000.

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

[RFC4181] Heard, C., "Guidelines for Authors and Reviewers of MIB Documents", BCP 111, RFC 4181, September 2005.

[RFC5153] Boschi, E., Mark, L., Quittek, J., Stiemerling, M., and P. Aitken, "IP Flow Information Export (IPFIX) Implementation Guidelines", RFC 5153, April 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.

[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.

[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.

[I-D.ietf-ipfix-structured-data] Claise, B., Dhandapani, G., Yates, S., and P. Aitken, "Export of Structured Data in IPFIX", draft-ietf-ipfix-structured-data-02 (work in progress), July 2010.

[I-D.ietf-ipfix-anon] Boschi, E. and B. Trammell, "IP Flow Anonymisation Support", draft-ietf-ipfix-anon-03 (work in progress), April 2010.

[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

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

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