

IPFIX Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 15th, 2013

P. Aitken
Cisco Systems
February 15, 2013

Reporting Equivalent IPFIX Information Elements
draft-aitken-ipfix-equivalent-ies-00

Abstract

This document proposes a method for IPFIX Exporting Processes to inform Collecting Processes of equivalent Information Elements, so that the Collecting Process can understand the equivalence and be enabled to process data across a change of Information Elements.

Status of this Memo

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

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

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

This Internet-Draft will expire on August 15, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Conventions used in this document

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

Table of Contents

1	Introduction	3
2	Terminology.....	4
3	Method	4
3.1	Equivalence Message Format	4
4	The Collecting Process's Side	6
5	Security Considerations	6
6	IANA Considerations	6
7	References	7
7.1	Normative References	7
7.2	Informative References	7
8	Acknowledgements	7
9	Author's Addresses	7

1 Introduction

The IPFIX Protocol [RFC5101] can export a large number of Information Elements, including standard elements specified in the IPFIX information model [RFC5102, IANA-IPFIX], Enterprise-Specific elements [RFC5101], and elements which are backwards compatible with NetFlow Version 9 [RFC3954].

From time to time, an Exporting Process may export the same information using different Information Elements from before.

Several scenarios (use cases) can be envisaged:

- . Enterprise-specific Information Elements have been standardised, so the Exporting Process is changed to export the IANA standard Information Elements [IANA-IPFIX] rather than the Enterprise-specific Information Elements.
- . The Exporting Process is changed to export IANA standard Information Elements [IANA-IPFIX] rather than NetFlow version 9 fields [RFC3954].
- . The Exporting Process is updated to export different Enterprise-specific Information Elements.
- . An updated Metering Process requests that the Exporting Process exports using different Information Elements from before.

In each case it's important to note that the same information is being exported. The only change is in the Information Element used to export the information.

Since different Information Elements are now being used to express the same information, the Collecting Process cannot process data received before the change with data received after the change, because the Collecting Process does not know that these Information Elements are related. e.g. it's not possible to compare, aggregate, or sort data across such a change without first understanding that the old and new Information Elements are equivalent.

Furthermore, it's impossible for every Collecting Process to know how each IANA standard Information Element [IANA-IPFIX] relates to every company's Enterprise-Specific Information Elements. ie, a Collecting Process from company X cannot be expected to know that company Y's Exporting Process exports Enterprise-specific field Z which is now equivalent to a certain IANA standard element.

This document proposes a method for Exporting Processes to inform Collecting Processes of such equivalence, so that the Collecting Process is able to process data across the change.

2 Terminology

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

3 Method

An Exporting Process informs a Collecting Process of the equivalence of a pair of IPFIX Information Elements by exporting an IPFIX Equivalence Message. Equivalence Messages SHOULD be sent by the Exporting Process upon opening a new Transport Session, before any other IPFIX Messages are exported. They may be sent in an Options Record Scoped to the Exporter. Multiple Equivalence Messages may be sent using IPFIX Structured Data [RFC6313].

3.1 Equivalence Message Format

The Equivalence Message consists of an original Information Element in the "informationElementId" field (#303), followed by the equivalent Information Element in the "equivalentElementId" field (#TBD), using the Template shown in Figure 1 and Data Record shown in Figure 2:

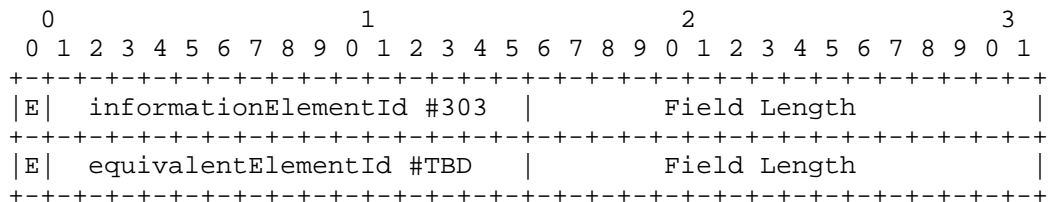


Figure 1: Template for Equivalence Message

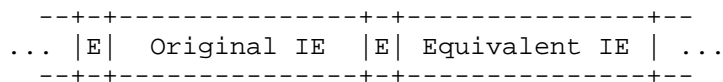


Figure 2: Equivalence Message Data Record

The encoding of these Information Elements follows the rules specified in [RFC5101].

When the original Information Element and the equivalent Information Element are both IANA standard elements [IANA-IPFIX], both of the E bits are zero and the Equivalence Message is as shown in Figure 1.

When the Original Information Element is Enterprise-Specific,

the Original Information Element's E bit is set and the Information Element number is immediately followed by the corresponding Private Enterprise Number [PEN], as shown in Figure 3:

```

+---+-----+-----+-----+-----+-----+-----+-----+
|1|  Original IE  |      Private Enterprise Number      |0| Equivalent IE |
+---+-----+-----+-----+-----+-----+-----+-----+

```

Figure 3: Equivalence Message with an Enterprise-Specific Original Information Element

This allows an Enterprise-Specific Information Element to be specified as equivalent to an IANA-standard Information Element.

When the Equivalent Information Element is Enterprise-Specific, the Equivalent Information Element's E bit is set and the Information Element number is immediately followed by the corresponding Private Enterprise Number [PEN] as shown in Figure 4:

```

+---+-----+-----+-----+-----+-----+-----+-----+
|0|  Original IE  |1| Equivalent IE  |      Private Enterprise Number      |
+---+-----+-----+-----+-----+-----+-----+-----+

```

Figure 4: Equivalence Message with an Enterprise Specific Equivalent Information Element

This allows an IANA-standard Information Element to be specified as equivalent to an Enterprise-Specific Information Element.

When both of the Information Elements are Enterprise-Specific, the E bits are set and both Information Element numbers are immediately followed by their corresponding Private Enterprise Number [PEN] as shown in Figure 5:

```

+---+-----+-----+-----+-----+-----+-----+-----+
|1|  Original IE  |      Private Enterprise Number      | ...
+---+-----+-----+-----+-----+-----+-----+-----+
... |1| Equivalent IE  |      Private Enterprise Number      |
+---+-----+-----+-----+-----+-----+-----+-----+

```

Figure 5: Equivalence Message with two Enterprise-Specific Information Elements

This allows two Enterprise-Specific Information Elements to be specified as equivalent.

Note that the Private Enterprise Numbers do not have to be equal. ie, the Information Elements may belong to different Private Enterprises.

4 The Collecting Process's Side

Equivalence Messages have global scope, unless they're sent in an Options Message with a more restrictive scope, eg an Options Record Scoped to the Exporter. ie, unless otherwise restricted, the specified equivalence applies to all devices.

Therefore the Collecting Process does not need to maintain equivalence per device.

5 Security Considerations

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

6 IANA Considerations

A new Information Element "equivalentElementId" must be allocated in IANA's IPFIX registry, [IANA-IPFIX]:

Description: An IPFIX Information Element ("equivalentElementId") that denotes an Information Element identifier which is equivalent to another Information Element identifier, specified in an IPFIX Equivalence Message [this document].

Abstract Data Type: Unsigned16

Data Type Semantics: identifier

ElementId: TBD

Status: current

Reference: [this document]

7 References

7.1 Normative References

- [RFC5101] Claise, B., Ed., "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information", RFC 5101, January 2008.
- [RFC5102] Quittek, J., Bryant, S., Claise, B., Aitken, P., and J. Meyer, "Information Model for IP Flow Information Export", RFC 5102, January 2008.
- [RFC3954] Claise, B., Ed., "Cisco Systems NetFlow Services Export Version 9", RFC 3954, October 2004.
- [RFC6313] Claise, B., Dhandapani, G., Aitken, P., and S. Yates, "Export of Structured Data in IP Flow Information Export (IPFIX)", RFC 6313, July 2011.
- [IANA-IPFIX] IANA, "IPFIX Information Elements registry", <<http://www.iana.org/assignments/ipfix/ipfix.xml>>.
- [PEN] IANA, "Private Enterprise Numbers registry", <<http://www.iana.org/assignments/enterprise-numbers>>.
- [RFC2119] S. Bradner, Key words for use in RFCs to Indicate Requirement Levels, BCP 14, RFC 2119, March 1997

7.2 Informative References

8 Acknowledgements

Thanks to you, dear reader.

9 Author's Addresses

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

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

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: August 27, 2013

S. D'Antonio
University of Napoli
"Parthenope"
T. Zseby
CAIDA/FhG FOKUS
C. Henke
Tektronix Communication Berlin
L. Peluso
University of Napoli
February 23, 2013

Flow Selection Techniques
draft-ietf-ipfix-flow-selection-tech-13.txt

Abstract

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

Requirements Language

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

Status of this Memo

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

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

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

This Internet-Draft will expire on August 27, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

Table of Contents

1. Scope	5
2. Terminology	5
3. Difference between Intermediate Flow Selection Process and Packet Selection	8
4. Intermediate Flow Selection Process within the IPFIX Architecture	9
4.1. Intermediate Flow Selection Process in the Metering Process	11
4.2. Intermediate Flow Selection Process in the Exporting Process	11
4.3. Intermediate Flow Selection Process as a function of the IPFIX Mediator	11
5. Intermediate Flow Selection Process Techniques	12
5.1. Flow Filtering	12
5.1.1. Property Match Filtering	12
5.1.2. Hash-based Flow Filtering	13
5.2. Flow Sampling	13
5.2.1. Systematic sampling	13
5.2.2. Random Sampling	14
5.3. Flow-state Dependent Intermediate Flow Selection Process	14
5.4. Flow-state Dependent Packet Selection	15
6. Configuration of Intermediate Flow Selection Process Techniques	15
6.1. Intermediate Flow Selection Process Parameters	17
6.2. Description of Flow-state Dependent Packet Selection	19
7. Information Model for Intermediate Flow Selection Process Configuration and Reporting	19
7.1. flowSelectorAlgorithm	21
7.2. flowSelectedOctetDeltaCount	22
7.3. flowSelectedPacketDeltaCount	22
7.4. flowSelectedFlowDeltaCount	22
7.5. selectorIDTotalFlowsObserved	23
7.6. selectorIDTotalFlowsSelected	23
7.7. samplingFlowInterval	24
7.8. samplingFlowSpacing	24
7.9. flowSamplingTimeInterval	24
7.10. flowSamplingTimeSpacing	25
7.11. hashFlowDomain	25
8. IANA Considerations	25
8.1. Registration of Information Elements	25
8.2. Registration of Object Identifier	34
9. Security Considerations	34
10. Acknowledgments	36
11. References	36
11.1. Normative References	36

11.2. Informative References	37
Appendix A. Appendix A. XML Specification of Intermediate Flow Selection Process Information Elements	38
Authors' Addresses	42

1. Scope

This document describes Intermediate Flow Selection Process techniques for network traffic measurements. A Flow is defined as a set of packets with common properties as described in [RFC5101]. An Intermediate Flow Selection Process can be executed to limit the resource demands for capturing, storing, exporting and post-processing of Flow Records. It also can be used to select a particular set of Flows that are of interest to a specific application. This document provides a categorization of Intermediate Flow Selection Process techniques and describes configuration and reporting parameters for them. In order to be compliant with this document, at least the Property Match Filtering MUST be implemented.

This document also addresses configuration and reporting parameters for Flow-state Dependent Packet Selection as described in [RFC5475], although this technique is categorized as packet selection. The reason is that Flow-state Dependent Packet Selection techniques often aim at the reduction of resources for Flow capturing and Flow processing. Furthermore, these techniques were only briefly discussed in [RFC5475]. Therefore configuration and reporting considerations for Flow-state Dependent Packet Selection techniques have been included in this document.

2. Terminology

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

* Packet Classification

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

* Intermediate Flow Selection Process

An Intermediate Flow Selection Process is an Intermediate Process as in [RFC6183] that takes Flow Records as its input and selects a subset of this set as its output. Intermediate Flow Selection Process is a more general concept than Intermediate Selection

Process as defined in [RFC6183]. While an Intermediate Selection Process selects Flow Records from a sequence based upon criteria-evaluated Flow record values and passes only those Flow Records that match the criteria, an Intermediate Flow Selection Process selects Flow Records using selection criteria applicable to a larger set of Flow characteristics and information.

* Flow Selection State

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

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

* Flow Selector

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

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

* Complete Flow

A Complete Flow consists of all the packets that enter the Intermediate Flow Selection Process within the Flow time-out interval, and which belong to the same Flow as defined by the Flow definition in [RFC5470]. For this definition only packets that arrive at the Intermediate Flow Selection Process are considered.

* Flow Filtering

Flow Filtering selects flows based on a deterministic function on the Flow Record content, Flow Selection State, external properties (e.g. ingress interface) or external events (e.g. violated Access Control List). If the relevant parts of the Flow Record content can already be observed at packet level (e.g. Flow Keys from packet header fields) Flow Filtering can be performed at packet level by Property Match Filtering as described in [RFC5475].

* Hash-based Flow Filtering

Hash-based Flow Filtering is a deterministic Flow filter function that selects flows based on a Hash Function. The Hash Function is calculated over parts of the Flow Record content or external properties which are called the Hash Domain. If the hash value falls into a predefined Hash Selection Range the Flow is selected. Hash-based Flow Filtering can already be applied at packet level, in which case the Hash Domain MUST contain the Flow Key of the packet. In case Hash-based Flow Filtering is used to select the same subset of flows at different Observation Points, the Hash Domain MUST comprise parts of the packet or Flow that are invariant on the packet/Flow path. Also refer to the according Trajectory Sampling Application Example on packet level in [RFC5475]

* Flow-state Dependent Intermediate Flow Selection Process

Flow-state Dependent Intermediate Flow Selection Process is a selection function that selects or drops Flows based on the current Flow Selection State. The selection can be either deterministic, random or non-uniform random.

* Flow-state Dependent Packet Selection

Flow-state Dependent Packet Selection is a selection function that selects or drops packets based on the current Flow Selection State. The selection can be either deterministic, random or non-uniform random. Flow-state Dependent Packet Selection can be used to prefer the selection of packets belonging to specific Flows. For example the selection probability of packets belonging to Flows that are already within the Flow Cache may be higher than for packets that have not been recorded yet.

* Flow Sampling

Flow Sampling selects flows based on Flow Record sequence or arrival times (e.g. entry in Flow cache, arrival time at Exporter or Mediator). The selection can be systematic (e.g. every n-th Flow) or based on a random function (e.g. select each Flow Record with probability p, or randomly select n out of N Flow Records).

3. Difference between Intermediate Flow Selection Process and Packet Selection

Intermediate Flow Selection Process differs from packet selection described in [RFC5475]. Packet selection techniques consider packets as the basic element and the parent population consists of all packets observed at an Observation Point. In contrast to this the basic elements in Flow selection are the Flows. The parent population consists of all observed Flows and the Intermediate Flow Selection Process operates on the Flows. The major characteristics of Intermediate Flow Selection Process are the following:

- Intermediate Flow Selection Process takes Flows as basic elements. For packet selection, packets are considered as basic elements.
- Intermediate Flow Selection Process can only take place after Packet Classification, because the classification rules determine to which Flow a packet belongs. Packet selection can be applied before and after Packet Classification. As an example, packet selection before Packet Classification can be random packet selection whereas packet selection after Packet Classification can be Flow-state Dependent Packet Selection (as described in [RFC5475])
- Intermediate Flow Selection Process operates on Complete Flows. That means that after the Intermediate Flow Selection Process either all packets of the Flow are kept or all packets of the Flow are discarded. That means that if the Intermediate Flow Selection Process is preceded by a packet selection process the Complete Flow consists only of the packets that were not discarded during the packet selection.

There are some techniques that are difficult to unambiguously categorize into one of the categories. Here some guidance is given on how to categorize such techniques:

- Techniques that can be considered as both packet selection and Intermediate Flow Selection Process: some packet selection techniques result in the selection of Complete Flows and therefore can be considered as packet selection or

as Intermediate Flow Selection Process at the same time. An example is Property Match Filtering of all packets to a specific destination address. If Flows are defined based on destination addresses, such a packet selection also results in a Intermediate Flow Selection Process and can be considered as packet selection or Intermediate Flow Selection Process.

- Flow-state Dependent Packet Selection: there exist techniques that select packets based on the Flow state, e.g. based on the number of already observed packets belonging to the Flow. Examples of these techniques from the literature are "Sample and Hold" [EsVa01] "Fast Filtered Sampling" [MSZC10] or the "Sticky Sampling" algorithm presented in [MaMo02]. Such techniques can be used to influence which Flows are captured (e.g. increase the selection of packets belonging to large Flows) and reduce the number of Flows that need to be stored in the Flow cache. Nevertheless, such techniques do not necessarily select Complete Flows, because they do not ensure that all packets of a selected Flow are captured. Therefore Flow-state Dependent Packet Selection techniques that do not ensure that either all or no packets of a Flow are selected strictly speaking have to be considered as packet selection techniques and not as Intermediate Flow Selection Process techniques.

4. Intermediate Flow Selection Process within the IPFIX Architecture

An Intermediate Flow Selection Process can be deployed at any of three places within the IPFIX architecture. As shown in Figure 1 Intermediate Flow Selection Process can occur

1. in the Metering Process at the IPFIX Exporter
2. in the Exporting Process at the Collector
3. within a Mediator

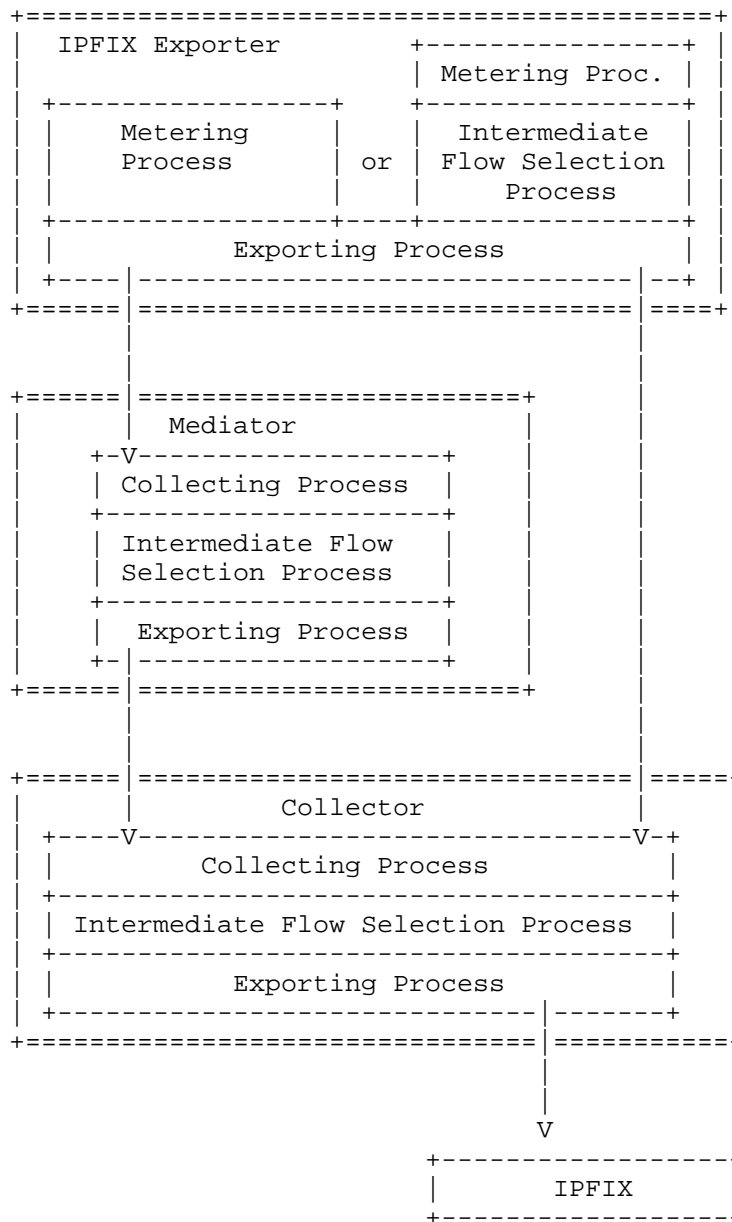


Figure 1: Potential Intermediate Flow Selection Process locations

In contrast to packet selection, Intermediate Flow Selection Process is always applied after the packets are classified into Flows.

4.1. Intermediate Flow Selection Process in the Metering Process

Intermediate Flow Selection Process in the Metering process uses packet information to update the Flow Records in the Flow cache. Intermediate Flow Selection Process before Packet Classification can be based on the fields of the Flow Key (also on a hash value over these fields), but not based on characteristics that are only available after Packet Classification (e.g. Flow size, Flow duration). An Intermediate Flow Selection Process is here applied to reduce resources for all succeeding processes or to select specific Flows of interest in case such Flow characteristics are already observable at packet level (e.g. Flows to specific IP addresses). In contrast, Flow-state Dependent Packet Selection is a packet selection technique, because it does not necessarily select Complete Flows.

4.2. Intermediate Flow Selection Process in the Exporting Process

Intermediate Flow Selection Process in the Exporting Process works on Flow Records. An Intermediate Flow Selection Process in the Exporting Process can therefore depend on Flow characteristics that are only visible after the classification of packets, such as Flow size and Flow duration. The Exporting Process may implement policies for exporting only a subset of the Flow Records which have been stored in the system memory in order to unload Flow export and Flow post-processing. An Intermediate Flow Selection Process in the Exporting Process may select only the subset of Flow Records which are of interest to the users application, or select only as many Flow Records as can be handled by the available resources (e.g. limited export link capacity).

4.3. Intermediate Flow Selection Process as a function of the IPFIX Mediator

As shown in Figure 1, Intermediate Flow Selection Process can be performed within an IPFIX Mediator [RFC6183]. The Intermediate Flow Selection Process takes Flow Record stream as its input and selects Flow Records from a sequence based upon criteria-evaluated record values. The Intermediate Flow Selection Process can again apply an Intermediate Flow Selection Process technique to obtain Flows of interest to the application. Further, the Intermediate Flow Selection Process can base its selection decision on the correlation of data from different IPFIX Exporters, e.g. by only selecting Flows that were at least recorded on two IPFIX Exporters.

5. Intermediate Flow Selection Process Techniques

An Intermediate Flow Selection Process technique selects either all or none of the packets of a Flow, otherwise the technique has to be considered as packet selection. A difference is recognized between Flow Filtering and Flow Sampling.

5.1. Flow Filtering

Flow Filtering is a deterministic function on the IPFIX Flow Record content. If the relevant Flow characteristics are already observable at packet level (e.g. Flow Keys), Flow Filtering can be applied before aggregation at packet level. In order to be compliant with this document, at least the Property Match Filtering MUST be implemented.

5.1.1. Property Match Filtering

Property Match Filtering can be performed similarly to Property Match Filtering for packet selection described in [RFC5475]. The difference is that, instead of packet fields, Flow Record fields are here used to derive the selection decision. Property Match Filtering is typically used to select a specific subset of the Flows that are of interest to a particular application (e.g. all Flows to a specific destination, all large Flows, etc.). Properties on which the filtering is based can be Flow Keys, Flow Timestamps, or Per-Flow Counters described in [RFC5102]. Examples of properties are the Flow size in bytes, the number of packets in the Flow, the observation time of the first or last packet, or the maximum packet length. An example is to select Flows with more than a threshold number of observed octets. The selection criteria can be a specific value, a set of specific values, or an interval. For example, a Flow is selected if destinationIPv4Address and the total number of packets of the Flow equal two predefined values. Property Match Filtering can be applied in the Metering Process if the properties are already observable at the packet level (e.g. Flow Key fields). For example, a Flow is selected if sourceIPv4Address and sourceIPv4PrefixLength equal, respectively, two specific values.

There are content-based Property Match Filtering techniques that require a computation on the current Flow cache. An example is the selection of the largest Flows or a percentage of Flows with the longest lifetime. This type of Property Match Filtering is also used in Intermediate Flow Selection Process techniques that react to external events (e.g. resource constraint). For example when the Flow cache is full, the Flow Record with the lowest Flow volume per current Flow life time may be deleted.

5.1.2. Hash-based Flow Filtering

Hash-based Flow Filtering uses a Hash Function h to map the Flow Key c onto a Hash Range R . A Flow is selected if the hash value $h(c)$ is within the Hash Selection Range S , which is a subset of R . Hash-based Flow Filtering can be used to emulate a random sampling process but still enable the correlation between selected Flow subsets at different Observation Points. Hash-based Flow Filtering is similar to Hash-based Packet Selection, and in fact is identical when Hash-based Packet Selection uses the Flow Key that defines the Flow as the hash input. Nevertheless there may be the incentive to apply Hash-based Flow Filtering not on the packet level in the Metering Process, for example when the size of the selection range and therefore the sampling probability is dependent on the number of observed Flows.

5.2. Flow Sampling

Flow Sampling operates on Flow Record sequence or arrival times. It can use either a systematic or a random function for the Intermediate Flow Selection Process. Flow Sampling usually aims at the selection of a representative subset of all Flows in order to estimate characteristics of the whole set (e.g. mean Flow size in the network).

5.2.1. Systematic sampling

Systematic sampling is a deterministic selection function. Systematic sampling may be a periodic selection of the N -th Flow Record which arrives at the Intermediate Flow Selection Process. Systematic sampling MAY be applied in the Metering Process. An example would be to create, besides the Flow cache of selected Flows, an additional data structure that saves the Flow Keys of the Flows that are not selected. The selection of a Flow would then be based on the first packet of a Flow. Everytime a packet belonging to a new Flow (which is neither in the data structure of the selected or not selected Flows) arrives at the Observation Point, a counter is increased. In case the counter is increased to a multiple of N a new Flow cache entry is created, and in case the counter is not a multiple of N the Flow Key is added to the data structure for not selected Flows.

Systematic sampling can also be time-based. Time-based systematic sampling is applied by only creating Flows that are observed between time-based start and stop triggers. The time interval may be applied at packet level in the Metering Process or after aggregation on Flow level, e.g. by selecting a Flow arriving at the Exporting Process every n seconds.

5.2.2. Random Sampling

Random Flow sampling is based on a random process which requires the calculation of random numbers. One can differentiate between n-out-N and probabilistic Flow sampling.

5.2.2.1. n-out-of-N Flow Sampling

In n-out-of-N Sampling, n elements are selected out of the parent population that consists of N elements. One example would be to generate n different random numbers in the range [1,N] and select all Flows that have a Flow position equal to one of the random numbers.

5.2.2.2. Probabilistic Flow Sampling

In probabilistic Sampling, the decision whether or not a Flow is selected is made in accordance with a predefined selection probability. For probabilistic Sampling, the Sample Size can vary for different trials. The selection probability does not necessarily have to be the same for each Flow. Therefore, a difference is recognized between uniform probabilistic sampling (with the same selection probability for all Flows) and non-uniform probabilistic sampling (where the selection probability can vary for different Flows). For non-uniform probabilistic Flow Sampling the sampling probability may be adjusted according to the Flow Record content. An example would be to increase the selection probability of large volume Flows over small volume Flows as described in the Smart Sampling technique [DuLT01].

5.3. Flow-state Dependent Intermediate Flow Selection Process

Flow-state Dependent Intermediate Flow Selection Process can be a deterministic or random Intermediate Flow Selection Process based on the Flow Record content and the Flow state which may be kept additionally for each of the Flows. External processes may update counters, bounds and timers for each of the Flow Records and the Intermediate Flow Selection Process utilises this information for the selection decision. A review of Flow-state Dependent Intermediate Flow Selection Process techniques that aim at the selection of the most frequent items by keeping additional Flow state information can be found in [CoHa08]. Flow-state Dependent Intermediate Flow Selection Process can only be applied after packet aggregation, when a packet has been assigned to a Flow. The Intermediate Flow Selection Process then decides based upon the Flow state for each Flow if it is kept in the Flow cache or not. Two Flow-state Dependent Intermediate Flow Selection Process Algorithms are here described:

The frequent algorithm [KaPS03] is a technique that aims at the selection of all flows that at least exceed a $1/k$ fraction of the Observed Packet Stream. The algorithm has only a Flow cache of size $k-1$ and each Flow in the cache has an additional counter. The counter is incremented each time a packet belonging to the Flow in the Flow cache is observed. In case the observed packet does not belong to any Flow all counters are decremented and if any of the Flow counters has a value of zero the Flow is replaced with a Flow formed from the new packet.

Lossy counting is a selection technique that identifies all Flows whose packet count exceeds a certain percentage of the whole observed packet stream (e.g. 5% of all packets) with a certain estimation error ϵ . Lossy counting separates the observed packet stream in windows of size $N=1/\epsilon$, where N is an amount of consecutive packets. For each observed Flow an additional counter will be held in the Flow state. The counter is incremented each time a packet belonging to the Flow is observed and all counters are decremented at the end of each window and all Flows with a counter of zero are removed from the Flow cache.

5.4. Flow-state Dependent Packet Selection

Flow-state Dependent Packet Selection is not an Intermediate Flow Selection Process technique but a packet selection technique. Nevertheless configuration and reporting parameters for this technique will be described in this document. An example is the "Sample and Hold" algorithm [EsVa01] that tries to prefer large volume Flows in the selection. When a packet arrives it is selected when a Flow Record for this packet already exists. In case there is no Flow Record, the packet is selected by a certain probability that is dependent on the packet size.

6. Configuration of Intermediate Flow Selection Process Techniques

This section describes the configuration parameters of the Flow selection techniques presented above. It provides the basis for an information model to be adopted in order to configure the Intermediate Flow Selection Process within an IPFIX Device. The actual information model with the Information Elements (IEs) for the configuration is described together with the reporting IEs in section 7. The following table gives an overview of the defined Intermediate Flow Selection Process techniques, where they can be applied and what their input parameters are. Depending on where the Flow selection techniques are applied different input parameters can be configured.

Overview of Intermediate Flow Selection Process Techniques:

Location	Selection Technique	Selection Input
In the Metering Process	Flow-state Dependent Packet Selection	packet sampling probabilities, Flow Selection State, packet properties
In the Metering Process	Property Match Flow Filtering	Flow record IEs, Selection Interval
In the Metering Process	Hash-based Flow Filtering	selection range, Hash Function, Flow Key, (seed)
In the Metering Process	Time-based Systematic Flow Sampling	Flow position (derived from arrival time of packets), Flow Selection State
In the Metering Process	Sequence-based Systematic Flow Sampling	Flow position (derived from packet position), Flow Selection State
In the Metering Process	Random Flow Sampling	random number generator or list and packet position, Flow state
In the Exporting Process/ within the IPFIX Mediator	Property Match Flow Filtering	Flow Record content, filter function
In the Exporting Process/ within the IPFIX Mediator	Hash-based Flow Filtering	selection range, Hash Function, hash input (Flow Keys and other Flow properties)
In the Exporting Process/ within the IPFIX Mediator	Flow-state Dependent Intermediate Flow Selection Process	Flow state parameters, random number generator or list
In the Exporting Process/ within the IPFIX Mediator	Time-based Systematic Flow Sampling	Flow arrival time, Flow state

In the Exporting Process/ within the IPFIX Mediator	Sequence-based Systematic Flow Sampling	Flow position, Flow state
In the Exporting Process/ within the IPFIX Mediator	Random Flow Sampling	random number generator or list and Flow position, Flow state

Table 1: Overview of Intermediate Flow Selection Process Techniques

6.1. Intermediate Flow Selection Process Parameters

This section defines what parameters are required to describe the most common Intermediate Flow Selection Process techniques.

Intermediate Flow Selection Process Parameters:

For Property Match Filtering:

- Information Element as specified in [iana-ipfix-assignments]): Specifies the Information Element which is used as the property in the filter expression.
- Selection Value or Value Interval: Specifies the value or interval of the filter expression. Packets and Flow Records that have a value equal to the Selection Value or within the Interval will be selected.

For Hash-based Flow Filtering:

- Hash Domain: Specifies the bits from the packet or Flow which are taken as the hash input to the Hash Function.
- Hash Function: Specifies the name of the Hash Function that is used to calculate the hash value. Possible Hash Functions are BOB [RFC5475], IPSX [RFC5475], CRC-32 [Bra75]
- Hash Selection Range: Flows that have a hash value within the Hash Selection Range are selected. The Hash Selection Range can be a value interval or arbitrary hash values within the Hash Range of the Hash Function.

- Random Seed or Initializer Value:
Some Hash Functions require an initializing value. In order to make the selection decision more secure one can choose a random seed that configures the hash function.

For Flow-state Dependent Intermediate Flow Selection Process:

- frequency threshold:
Specifies the frequency threshold s for Flow-state Dependent Flow Selection techniques that try to find the most frequent items within a dataset. All Flows which exceed the defined threshold will be selected.
- accuracy parameter:
specifies the accuracy parameter e for techniques that deal with the frequent items problems. The accuracy parameter defines the maximum error, i.e. no Flows that have a true frequency less than $(s - e)N$ are selected, where s is the frequency threshold and N is the total number of packets.

The above list of parameters for Flow-state Dependent Flow Selection techniques is suitable for the presented frequent item and lossy counting algorithms. Nevertheless a variety of techniques exist with very specific parameters which are not defined here.

For Systematic time-based Flow Sampling:

- Interval length (in usec)
Defines the length of the sampling interval during which Flows are selected.
- Spacing (in usec)
The spacing parameter defines the spacing in usec between the end of one sampling interval and the start of the next succeeding interval.

For Systematic count-based Flow Sampling:

- Interval length
Defines the number of Flows that are selected within the sampling interval.
- Spacing
The spacing parameter defines the spacing in number of observed Flows between the end of one sampling interval and the start of the next succeeding interval.

For random n-out-of-N Flow Sampling:

- Population Size N
The Population Size N is the number of all Flows in the Population from which the sample is drawn.
- Sampling Size n
The sampling size n is the number of Flows that are randomly drawn from the population N.

For probabilistic Flow Sampling:

- Sampling probability p
The sampling probability p defines the probability by which each of the observed Flows is selected.

6.2. Description of Flow-state Dependent Packet Selection

The configuration of Flow-state Dependent Packet Selection has not been described in [RFC5475] therefore the parameters are defined here:

For Flow-state Dependent Packet Selection:

- packet selection probability per possible Flow state interval
Defines multiple {Flow interval, packet selection probability} value pairs that configure the sampling probability depending on the current Flow state.
- additional parameters
For the configuration of Flow-state Dependent Packet Selection additional parameters or packet properties may be required, e.g. the packet size ([EsVa01])

7. Information Model for Intermediate Flow Selection Process Configuration and Reporting

This section specifies the Information Elements (IEs) that MUST be exported by an Intermediate Flow Selection Process in order to support the interpretation of measurement results from Flow measurements. The information is mainly used to report how many packets and Flows have been observed in total and how many of them were selected. This helps for instance to calculate the Attained Selection Fraction (see also [RFC5476]), which is an important parameter to provide an accuracy statement. The IEs can provide reporting information about Flow Records, packets or bytes. The reported metrics are total number of elements and the number of selected elements. From this the number of dropped elements can be derived.

List of Intermediate Flow Selection Process Information Elements:

ID	Name	ID	Name
301	selectionSequenceID	302	selectorID
TBD 1	flowSelectorAlgorithm	1	octetDeltaCount
TBD 2	flowSelectedOctetDeltaCount	2	packetDeltaCount
TBD 3	flowSelectedPacketDeltaCount	3	originalFlowsPresent
TBD 4	flowSelectedFlowDeltaCount	TBD5	selectorIDTotalFlowsObserved
TBD 6	selectorIDTotalFlowsSelected	TBD7	samplingFlowInterval
TBD 8	samplingFlowSpacing	309	samplingSize
310	samplingPopulation	311	samplingProbability
TBD 9	flowSamplingTimeInterval	TBD10	flowSamplingTimeSpacing
326	digestHashValue	TBD11	hashFlowDomain
329	hashOutputRangeMin	330	hashOutputRangeMax
331	hashSelectedRangeMin	332	hashSelectedRangeMax
333	hashDigestOutput	334	hashInitialiserValue
320	absoluteError	321	relativeError
336	upperCILimit	337	lowerCILimit
338	confidenceLevel		

Table 2: Intermediate Flow Selection Process Information Elements

7.1. flowSelectorAlgorithm

Description:

This Information Element identifies the Intermediate Flow Selection Process technique(e.g., Filtering, Sampling) that is applied by the Intermediate Flow Selection Process. Most of these techniques have parameters as described in Section 6. Further technique identifiers may be added to the list below. It might be necessary to define new Information Elements to specify their parameters. The flowSelectorAlgorithm registry is maintained by IANA. New assignments for the registry will be administered by IANA and are subject to Expert Review [RFC5226]. The registry can be updated when specifications of the new technique(s) and any new Information Elements are provided.

ID	Technique	Parameters
1	Systematic count-based Sampling	flowSamplingInterval flowSamplingSpacing
2	Systematic time-based Sampling	flowSamplingTimeInterval flowSamplingTimeSpacing
3	Random n-out-of-N Sampling	samplingSize samplingPopulation
4	Uniform probabilistic Sampling	samplingProbability
5	Property Match Filtering	Information Element Value Range
	Hash-based Filtering	hashInitialiserValue hashFlowDomain
6	using BOB	hashSelectedRangeMin hashSelectedRangeMax
7	using IPSX	hashOutputRangeMin hashOutputRangeMax
8	using CRC	
TBDx	Flow-state Dependent Intermediate Flow Selection Process	No agreed Parameters

Intermediate Flow Selection Process Techniques

Abstract Data Type: unsigned16

ElementId: TBD1

Data Type Semantics: identifier

Status: Proposed

7.2. flowSelectedOctetDeltaCount

Description:

This Information Element specifies the volume in octets of all Flows that are selected in the Intermediate Flow Selection Process since the previous report.

Abstract Data Type: unsigned64

ElementId: TBD2

Units: Octets

Status: Proposed

7.3. flowSelectedPacketDeltaCount

Description:

This Information Element specifies the volume in packets of all Flows that were selected in the Intermediate Flow Selection Process since the previous report.

Abstract Data Type: unsigned64

ElementId: TBD3

Units: Packets

Status: Proposed

7.4. flowSelectedFlowDeltaCount

Description:

This Information Element specifies the number of Flows that were selected in the Intermediate Flow Selection Process since the last

report.

Abstract Data Type: unsigned64

ElementId: TBD4

Units: Flows

Status: Proposed

7.5. selectorIDTotalFlowsObserved

Description:

This Information Element specifies the total number of Flows observed by a Selector, for a specific value of SelectorId. This Information Element should be used in an Options Template scoped to the observation to which it refers. See Section 3.4.2.1 of the IPFIX protocol document [RFC5101] .

Abstract Data Type: unsigned64

ElementId: TBD5

Units: Flows

Status: Proposed

7.6. selectorIDTotalFlowsSelected

Description:

This Information Element specifies the total number of Flows selected by a Selector, for a specific value of SelectorId. This Information Element should be used in an Options Template scoped to the observation to which it refers. See Section 3.4.2.1 of the IPFIX protocol document [RFC5101].

Abstract Data Type: unsigned64

ElementId: TBD6

Units: Flows

Status: Proposed

7.7. samplingFlowInterval

Description:

This Information Element specifies the number of Flows that are consecutively sampled. A value of 100 means that 100 consecutive Flows are sampled. For example, this Information Element may be used to describe the configuration of a systematic count-based Sampling Selector.

Abstract Data Type: unsigned64

ElementId: TBD7

Units: Flows

Status: Proposed

7.8. samplingFlowSpacing

Description:

This Information Element specifies the number of Flows between two "samplingFlowInterval"s. A value of 100 means that the next interval starts 100 Flows (which are not sampled) after the current "samplingFlowInterval" is over. For example, this Information Element may be used to describe the configuration of a systematic count-based Sampling Selector.

Abstract Data Type: unsigned64

ElementId: TBD8

Units: Flows

Status: Proposed

7.9. flowSamplingTimeInterval

Description:

This Information Element specifies the time interval in microseconds during which all arriving Flows are sampled. For example, this Information Element may be used to describe the configuration of a systematic time-based Sampling Selector.

Abstract Data Type: unsigned64

ElementId: TBD9

Units: microseconds

Status: Proposed

7.10. flowSamplingTimeSpacing

Description:

This Information Element specifies the time interval in microseconds between two "flowSamplingTimeInterval"s. A value of 100 means that the next interval starts 100 microseconds (during which no Flows are sampled) after the current "flowsamplingTimeInterval" is over. For example, this Information Element may be used to describe the configuration of a systematic time-based Sampling Selector.

Abstract Data Type: unsigned64

ElementId: TBD10

Units: microseconds

Status: Proposed

7.11. hashFlowDomain

Description:

This Information Element specifies the Information Elements that are used by the Hash-based Flow Selector as the Hash Domain.

Abstract Data Type: unsigned16

ElementId: TBD11

Data Type Semantics: identifier

Status: Proposed

8. IANA Considerations

8.1. Registration of Information Elements

IANA will register the following IEs in the IPFIX Information Elements registry at <http://www.iana.org/assignments/ipfix/ipfix.xml>

Value	Name	Data Type	Data Type Semantics	Status	Description
TBD 1	flowSelectorAlgorithm	unsigned16	identifier	Proposed	This Information Element identifies the Intermediate Flow Selection Process technique(e.g., Filtering, Sampling) that is applied by the Intermediate Flow Selection Process
TBD 2	flowSelectedOctetDeltaCount	unsigned64	Octets	Proposed	This Information Element specifies the volume in octets of all Flows that are selected in the Intermediate Flow Selection Process since the previous report.

TBD 3	flowSelectedPacketDeltaCount	unsigned64	Packets	Proposed	This Information Element specifies the volume in packets of all Flows that were selected in the Intermediate Flow Selection Process since the previous report.
TBD 4	flowSelectedFlowDeltaCount	unsigned64	Flows	Proposed	This Information Element specifies the number of Flows that were selected in the Intermediate Flow Selection Process since the last report.

TBD 5	selectorIDTotalFlowsObserved	unsigned64	Flows	Proposed	This Information Element specifies the total number of Flows observed by a Selector, for a specific value of SelectorId. This Information Element should be used in an Options Template scoped to the observation to which it refers. See Section 3.4.2.1 of the IPFIX protocol document [RFC5101]
----------	------------------------------	------------	-------	----------	--

TBD 6	selectorIDTotalFlowsSelected	unsigned64	Flows	Proposed	This Information Element specifies the total number of Flows selected by a Selector, for a specific value of SelectorId. This Information Element should be used in an Options Template scoped to the observation to which it refers. See Section 3.4.2.1 of the IPFIX protocol document [RFC5101].
----------	------------------------------	------------	-------	----------	---

TBD 7	samplingFlowIn terval	unsign ed64	Flows	Propo sed	This Information Element specifies the number of Flows that are consecutively sampled. A value of 100 means that 100 consecutive Flows are sampled. For example, this Information Element may be used to describe the configuration of a systematic count-based Sampling Selector.
----------	--------------------------	----------------	-------	--------------	--

TBD 8	samplingFlowSp acing	unsign ed64	Flows	Propo sed	This Information Element specifies the number of Flows between two "samplingFlowIn terval"s. A value of 100 means that the next interval starts 100 Flows (which are not sampled) after the current "samplingFlowI nterval" is ove r. For example, this Information Element may b e used to describe the configuration of a systemat iccount-based Sampling Selector.
----------	-------------------------	----------------	-------	--------------	---

TBD 9	flowSamplingTimeInterval	unsigned64	microseconds	Proposed	This Information Element specifies the time interval in microseconds during which all arriving Flows are sampled. For example, this Information Element may be used to describe the configuration of a systematic time-based Sampling Selector.
----------	--------------------------	------------	--------------	----------	---

TBD 10	flowSamplingTimeSpacing	unsigned64	microseconds	Proposed	This Information Element specifies the time interval in microseconds between two "flowSamplingTimeInterval"s. A value of 100 means that the next interval starts 100 microseconds (during which no Flows are sampled) after the current "flowSamplingTimeInterval" is over. For example, this Information Element may be used to describe the configuration of a systematic time-based Sampling Selector.
TBD 11	hashFlowDomain	unsigned16	identifier	Proposed	This Information Element specifies the Information Elements that are used by the Hash-based Flow Selector as the Hash Domain.

Table 3: Information Elements to be registered

IANA Note: please replace TBD1, TBD2, TBD3, TBD4, TBD5, TBD6, TBD7,

TBD8, TBD9, TBD10, TBD11 with the assigned values, throughout the document

8.2. Registration of Object Identifier

IANA will register the following OID in the IPFIX-SELECTOR-MIB Functions sub-registry at <http://www.iana.org/assignments/smi-numbers> according to the procedures set forth in [RFC6615]

Decimal	Name	Description	Reference
	flowSelectorAlgorithm	This Object Identifier identifies the Intermediate Flow Selection Process technique (e.g., Filtering, Sampling) that is applied by the Intermediate Flow Selection Process	TBDx [RFCyyyy]

Table 4: Object Identifiers to be registered

IANA Note: please replace TBDx with the assigned value, throughout the document.

Editor's Note (to be removed prior to publication): the RFC editor is asked to replace "yyyy" in this document by the number of the RFC when the assignment has been made.

9. Security Considerations

Some of the described Intermediate Flow Selection Process techniques (e.g., flow sampling, hash-based flow filtering) aim at the selection of a representative subset of flows in order to estimate parameters of the population. An adversary may have incentives to influence the selection of flows, for example to circumvent accounting or to avoid the detection of packets that are part of an attack.

Security considerations concerning the choice of a Hash Function for Hash-based Packet Selection have been discussed in Section 6.2.3 of [RFC5475] and are also appropriate for Hash-based Flow Selection. [RFC5475] discusses the possibility to craft Packet Streams which are

disproportionately selected or can be used to discover Hash Function parameters. It also describes vulnerabilities of different Hash Functions to these attacks, and practices to minimize these vulnerabilities.

For other sampling approaches a user can gain knowledge about the start and stop triggers in time-based systematic Sampling, e.g., by sending test packets. This knowledge might allow users to modify their send schedule in a way that their packets are disproportionately selected or not selected. For random Sampling, an input to the encryption process, like the Initialization Vector of the CBC (Cipher Block Chaining) mode, should be used to prevent that an adversary can predict the selection decision [Dw01].

Further security threats can occur when Intermediate Flow Selection Process parameters are configured or communicated to other entities. The protocol(s) for the configuration and reporting of Intermediate Flow Selection Process parameters are out of scope of this document. Nevertheless, a set of initial requirements for future configuration and reporting protocols are stated below:

1. Protection against disclosure of configuration information: Intermediate Flow Selection Process configuration information describes the Intermediate Flow Selection Process and its parameters. This information can be useful to attackers. Attackers may craft packets that never fit the selection criteria in order to prevent Flows to be seen by the Intermediate Flow Selection Process. They can also craft a lot of packets that fit the selection criteria and overload or bias subsequent processes. Therefore any transmission of configuration data (e.g., to configure a process or to report its actual status) should be protected by encryption.
2. Protection against modification of configuration information: if wrong configuration information is sent to the Intermediate Flow Selection Process, it can lead to a malfunction of the Intermediate Flow Selection Process. Also if wrong configuration information is reported from the Intermediate Flow Selection Process to other processes it can lead to wrong estimations at subsequent processes. Therefore any protocol that transmits configuration information should prevent that an attacker can modify configuration information. Data integrity can be achieved by authenticating the data.
3. Protection against malicious nodes sending configuration information: The remote configuration of Intermediate Flow Selection Process techniques should be protected against access by unauthorized nodes. This can be achieved by access control

lists at the device that hosts the Intermediate Flow Selection Process (e.g. IPFIX Exporter, IPFIX Mediator or IPFIX Collector) and by source authentication. The reporting of configuration data from an Intermediate Flow Selection Process has to be protected in the same way. That means that also protocols that report configuration data from the Intermediate Flow Selection Process to other processes need to protect against unauthorized nodes reporting configuration information.

The security threats that originate from communicating configuration information to and from Intermediate Flow Selection Processes cannot be assessed solely with the information given in this document. A further more detailed assessment of security threats is necessary when a specific protocol for the configuration or reporting configuration data is proposed.

10. Acknowledgments

We would like to thank the IPFIX group, especially Brian Trammell, Paul Aitken and Benoit Claise for fruitful discussions and for proofreading the document.

11. References

11.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC5101] Claise, B., "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information", RFC 5101, January 2008.
- [RFC5102] Quittek, J., Bryant, S., Claise, B., Aitken, P., and J. Meyer, "Information Model for IP Flow Information Export", RFC 5102, January 2008.
- [RFC5475] Zseby, T., Molina, M., Duffield, N., Niccolini, S., and F. Raspall, "Sampling and Filtering Techniques for IP Packet Selection", RFC 5475, March 2009.
- [RFC5476] Claise, B., Johnson, A., and J. Quittek, "Packet Sampling (PSAMP) Protocol Specifications", RFC 5476, March 2009.
- [RFC6615] Dietz, T., Kobayashi, A., Claise, B., and G. Muenz, "Definitions of Managed Objects for IP Flow Information

Export", RFC 6615, June 2012.

11.2. Informative References

- [Bra75] Brayer, K., "Evaluation of 32 Degree Polynomials in Error Detection on the SATIN IV Auto von Error Patterns", National Technical Information Service p.74, August 1975.
- [CoHa08] Cormode, G. and M. Hadjieleftheriou, "Finding frequent items in data streams", Journal, Proceedings of the Very Large DataBase Endowment VLDB Endowment, Volume 1 Issue 2, August 2008, August 2008.
- [DuLT01] Duffield, N., Lund, C., and M. Thorup, "Charging from Sampled Network Usage", ACM Internet Measurement Workshop IMW 2001, San Francisco, USA, November 2001.
- [Dw01] Dworkin, M., "Recommendation for Block Cipher Modes of Operation - Methods and Techniques", NIST Special Publication NIST Special Publication 800-38A 2001 Edition, December 2001.
- [EsVa01] Estan, C. and G. Varghese, "New Directions in Traffic Measurement and Accounting: Focusing on the Elephants, Ignoring the Mice", ACM SIGCOMM Internet Measurement Workshop 2001, San Francisco (CA), November 2001.
- [KaPS03] Karp, R., Papadimitriou, C., and S. S. Shenker, "A simple algorithm for finding frequent elements in sets and bags.", ACM Transactions on Database Systems, Volume 28, 51-55, 2003, March 2003.
- [MSZC10] Mai, J., Sridharan, A., Zang, H., and C. Chuah, "Fast Filtered Sampling", Computer Networks Volume 54, Issue 11, Pages 1885-1898, ISSN 1389-1286, January 2010.
- [MaMo02] Manku, G. and R. Motwani, "Approximate Frequency Counts over Data Streams", Proceedings of the International Conference on Very large DataBases (VLDB) pages 346--357, 2002, Hong Kong, China, 2002.
- [RFC3917] Quittek, J., Zseby, T., Claise, B., and S. Zander, "Requirements for IP Flow Information Export (IPFIX)", RFC 3917, October 2004.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

- [RFC5470] Sadasivan, G., Brownlee, N., Claise, B., and J. Quittek, "Architecture for IP Flow Information Export", RFC 5470, March 2009.
- [RFC6183] Kobayashi, A., Claise, B., Muenz, G., and K. Ishibashi, "IP Flow Information Export (IPFIX) Mediation: Framework", RFC 6183, April 2011.
- [iana-ipfix-assignments] "IP Flow Information Export Information Elements", 2007, <<http://www.iana.org/assignments/ipfix/ipfix.xml>>.

Appendix A. Appendix A. XML Specification of Intermediate Flow Selection Process Information Elements

This appendix contains a machine-readable description of the Intermediate Flow Selection Process Information Elements coded in XML. Note that this appendix is of informational nature, while the text in Section 7 is normative. The format in which this specification is given is described by the XML Schema in Appendix B of [RFC5102].

```

o"
    <fieldDefinitions xmlns="urn:ietf:params:xml:ns:ipfix-inf
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="urn:ietf:params:xml:ns:ipfix-info
    ipfix-info.xsd">
        <field name="flowSelectorAlgorithm" dataType="unsigned16"
        dataTypeSemantics="identifier"
        elementId="TBD1" status="proposed">
            <description>
                <paragraph>
                    This Information Element identifies the Intermedi
                    ate Flow Selection Process technique(e.g., Filtering, Sampling) that is applied b
                    y the Intermediate Flow Selection Process. Most of these techniques have paramete
                    rs as decribed in <xref target="config"/>. Further technique identifiers may be a
                    dded to the list below. It might be necessary to define new Information Elements
                    to specify their parameters. The flowSelectorAlgorithm registry is maintained by
                    IANA. New assignments for the registry will be administered by IANA and are subje
                    ct to Expert Review <xref target="RFC5226"/>. The registry can be updated when sp
                    ecifications of the new technique(s) and any new Information
                    Elements are provided.
                </paragraph>
                <artwork>
                    +-----+-----+-----+-----+
                    | ID |           Technique           |           Parameters           |
                    +-----+-----+-----+-----+
                    | 1 | Systematic count-based | flowSamplingInter
                    |   | Sampling                | flowSamplingSpaci
                    +-----+-----+-----+-----+

```

-----+			
interval	2	Systematic time-based	flowSamplingTimeI
pacing		Sampling	flowSamplingTimeS
-----+	+-----+	-----+	-----+
	3	Random n-out-of-N	samplingSize

n			Sampling	samplingPopulatio
		+-----+	+-----+	+-----+
ty		4	Uniform probabilistic	samplingProbabili
			Sampling	
nt		+-----+	+-----+	+-----+
		5	Property Match	Information Eleme
lue			Filtering	Value Range
		+-----+	+-----+	+-----+
Min			Hash-based Filtering	hashInitialiserVa
		+-----+	+-----+	+ hashFlowDomain
Max		6	using BOB	hashSelectedRange
		+-----+	+-----+	+ hashSelectedRange
n		7	using IPSX	hashOutputRangeMi
		+-----+	+-----+	+ hashOutputRangeMa
x		8	using CRC	
		+-----+	+-----+	+-----+
ers		TBDx	Flow-state Dependent	No agreed Paramet
			Intermediate Flow	
			Selection Process	
		+-----+	+-----+	+-----+

</artwork>

</description>

</field>

```
<field name="flowSelectedOctetDeltaCount" dataType="unsigned64"
  elementId="TBD2" status="proposed">
```

<description>

<paragraph>

octets of all

This Information Element specifies the volume in

Selection Process

Flows that are selected in the Intermediate Flow

since the previous report.

</paragraph>

</description>

<units>octets</units>

</field>


```

        <field name="flowSelectedPacketDeltaCount" dataType="unsigned64"
elementId="TBD3" status="proposed">
        <description>
            <paragraph>
                This Information Element specifies the volume in
packets of all
                Flows that were selected in the Intermediate Flow
                Selection
                Process since the previous report.
            </paragraph>
        </description>
        <units>packets</units>
    </field>

```

```

    <field name="flowSelectedFlowDeltaCount" dataType="unsigned64"
      elementId="TBD4" status="proposed">
      <description>
        <paragraph>
          This Information Element specifies the number of
Flows that were
          selected in the Intermediate Flow Selection Proce
ss since the last
          report.
        </paragraph>
      </description>
      <units>flows</units>
    </field>

    <field name="selectorIDTotalFlowsObserved" dataType="unsigned64"
      elementId="TBD5" status="proposed">
      <description>
        <paragraph>
          This Information Element specifies the total numb
er of Flows
          observed by a Selector, for a specific value of S
selectorId. This
          Information Element should be used in an Options
Template scoped
          to the observation to which it refers.
        </paragraph>
      </description>
      <reference>
        <paragraph>
          See Section 3.4.2.1 of the IPFIX protocol documen
t <xref target="RFC5101"/>
        </paragraph>
      </reference>
      <units>flows</units>
    </field>

    <field name="selectorIDTotalFlowsSelected" dataType="unsigned64"
      elementId="TBD6" status="proposed">
      <description>
        <paragraph>
          This Information Element specifies the total numb
er of Flows
          selected by a Selector, for a specific value of S
selectorId. This
          Information Element should be used in an Options
Template scoped
          to the observation to which it refers.
        </paragraph>
      </description>
      <reference>
        <paragraph>
          See Section 3.4.2.1 of the IPFIX protocol documen
t <xref target="RFC5101"/>
        </paragraph>
      </reference>
      <units>flows</units>
    </field>

```



```
<field name="samplingFlowInterval" dataType="unsigned64"
  elementId="TBD7" status="proposed">
  <description>
    <paragraph>
      This Information Element specifies the number of
    </paragraph>
  </description>
  <units>flows</units>
</field>
```

Flows that are
100 consecutive
Element may be
c count-based

```
<field name="samplingFlowSpacing" dataType="unsigned64"
  elementId="TBD8" status="proposed">
  <description>
    <paragraph>
      This Information Element specifies the number of
    </paragraph>
  </description>
  <units>flows</units>
</field>
```

Flows between two
at the next
after the
le, this
onfiguration of a

```
<field name="flowSamplingTimeInterval" dataType="unsigned64"
  elementId="TBD9" status="proposed">
  <description>
    <paragraph>
      This Information Element specifies the time inter
    </paragraph>
  </description>
  <units>microseconds</units>
</field>
```

val in
sampled. For
describe the
Selector.

```
<field name="flowSamplingTimeSpacing" dataType="unsigned64"
  elementId="TBD10" status="proposed">
  <description>
```

microseconds during which all arriving Flows are
example, this Information Element may be used to
configuration of a systematic time-based Sampling

<paragraph>
This Information Element specifies the time inter
val in
microseconds between two "flowSamplingTimeInterva
l"s. A value of

D'Antonio, et al.

Expires August 27, 2013

[Page 41]

100 means that the next interval starts 100 micro
seconds (during
which no Flows are sampled) after the current
"flowsamplingTimeInterval" is over. For example,
this Information
Element may used to describe the configuration of
a systematic
time-based Sampling Selector.
</paragraph>
</description>
<units>microseconds</units>
</field>

<field name="hashFlowDomain" dataType="unsigned16"
dataTypeSemantics="identifier"
elementId="TBD11" status="proposed">
<description>
<paragraph>
This Information Element specifies the Informatio
n Elements that
are used by the Hash-based Flow Selection Selecto
r as the Hash
Domain.
</paragraph>
</description>
</field>

Authors' Addresses

Salvatore D'Antonio
University of Napoli "Parthenope"
Centro Direzionale di Napoli Is. C4
Naples 80143
Italy

Phone: +39 081 5476766
Email: salvatore.dantonio@uniparthenope.it

Tanja Zseby
CAIDA/FhG FOKUS
San Diego Supercomputer Center (SDSC)
University of California, San Diego (UCSD)
9500 Gilman Drive
La Jolla CA 92093-0505
USA

Email: tanja@caida.org

Christian Henke
Tektronix Communication Berlin
Wohlrabedamm 32
Berlin 13629
Germany

Phone: +49 17 2323 8717
Email: christian.henke@tektronix.com

Lorenzo Peluso
University of Napoli
Via Claudio 21
Napoli 80125
Italy

Phone: +39 081 7683821
Email: lorenzo.peluso@unina.it

IPFIX Working Group
Internet-Draft
Intended status: BCP
Expires: April 6, 2013

B. Trammell
ETH Zurich
B. Claise
Cisco Systems, Inc.
October 3, 2012

Guidelines for Authors and Reviewers of IPFIX Information Elements
draft-ietf-ipfix-ie-doctors-07.txt

Abstract

This document provides guidelines for how to write definitions of new Information Elements for the IP Flow Information Export (IPFIX) protocol. It provides instructions on using the proper conventions for Information Elements to be registered in the IANA IPFIX Information Element registry, and provides guidelines for expert reviewers to evaluate new registrations.

Status of this Memo

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

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

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

This Internet-Draft will expire on April 6, 2013.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	4
1.1. Intended Audience and Usage	4
1.2. Overview of relevant IPFIX documents	5
2. Terminology	5
3. How to apply IPFIX	6
4. Defining new Information Elements	7
4.1. Information Element naming	8
4.2. Information Element data types	8
4.3. Information Element numbering	9
4.4. Ancillary Information Element properties	10
4.5. Internal structure in Information Elements	10
4.6. Information Element multiplicity	11
4.7. Enumerated Values and Subregistries	12
4.8. Reversibility as per RFC 5103	12
4.9. Avoiding Bad Ideas in Information Element Design	13
5. The Information Element Lifecycle	14
5.1. The IE-DOCTORS process	14
5.2. Revising Information Elements	15
5.3. Deprecating Information Elements	16
6. When not to define new Information Elements	17
6.1. Maximizing reuse of existing Information Elements	17
6.2. Applying enterprise-specific Information Elements	19
7. Information Element Definition Checklist	19
8. Applying IPFIX to non-Flow Applications	22
9. Writing Internet-Drafts for IPFIX Applications	22
9.1. Example Information Element Definition	23
9.2. Defining Recommended Templates	23
10. A Textual Format for Specifying Information Elements and Templates	24
10.1. Information Element Specifiers	25
10.2. Specifying Templates	27
10.3. Specifying IPFIX Structured Data	28
11. Security Considerations	28
12. IANA Considerations	29
13. Acknowledgements	29
14. References	30
14.1. Normative References	30
14.2. Informative References	30
Appendix A. Example Information Element Definitions	31
A.1. sipResponseStatus	32
A.2. duplicatePacketDeltaCount	32
A.3. ambientTemperature	33

Authors' Addresses	33
------------------------------	----

1. Introduction

This document provides guidelines for the definition of new IPFIX Information Elements beyond those currently in the IANA IPFIX Information Element Registry [iana-ipfix-assignments]. Given the self-describing nature of the data export format used by IPFIX, the definition of new Information Elements is often sufficient to allow the application of IPFIX to new network measurement and management use cases.

We intend this document to enable the application of IPFIX to new areas by experts in the IETF Working Group or Area Directorate, or the community or organization external to the IETF, concerned with the technical details of the protocol or application to be measured or managed using IPFIX. This expansion occurs 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.

This document essentially codifies two meta-guidelines: (1) "define new Information Elements that look like existing Information Elements", and (2) "don't define Information Elements unless you need to".

1.1. Intended Audience and Usage

This document is meant for two separate audiences. For those defining new Information Elements, it provides specifications and best practices to be used in deciding which Information Elements are necessary for a given existing or new application, instructions for writing the definitions for these Information Elements, and information on the supporting documentation required for the new application (up to and including the publication of one or more RFCs describing it). For the IPFIX experts appointed as IE-DOCTORS, and for IANA personnel changing the IANA IPFIX Information Element Registry [iana-ipfix-assignments], 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 within the IPFIX Working Group.

This document, together with [I-D.ietf-ipfix-information-model-rfc5102bis], defines the procedures for management of the IANA IPFIX Information Element Registry [iana-ipfix-assignments]. The practices outlined in this document

are intended to guide experts when reviewing additions or changes to the Information Elements in the registry under Expert Review as defined in [RFC5226].

1.2. Overview of relevant IPFIX documents

[I-D.ietf-ipfix-protocol-rfc5101bis] defines the IPFIX Protocol, the IPFIX-specific terminology used by this document, and the data type encodings for each of the data types supported by IPFIX.

[I-D.ietf-ipfix-information-model-rfc5102bis] defines the basis of the IPFIX Information Model, referring to [iana-ipfix-assignments] for the specific Information Element definitions. It states that new Information Elements may be added to the Information Model on Expert Review basis, delegates the appointment of experts to an IESG Area Director, and refers to this document for details on the extension process. This document is intended to further codify the best practices to be followed by these experts, in order to improve the efficiency of this process.

[RFC5103] defines a method for exporting bidirectional flow information using IPFIX; this document should be followed when extending IPFIX to represent information about bidirectional network interactions in general. Additionally, new Information Elements should be annotated for their reversibility or lack thereof as per this document.

[RFC5610] defines a method for exporting information about Information Elements inline within IPFIX. In doing so, it explicitly defines a set of restrictions, implied in [I-D.ietf-ipfix-protocol-rfc5101bis] and [I-D.ietf-ipfix-information-model-rfc5102bis], on the use of data types and semantic; these restrictions must be observed in the definition of new Information Elements, as in Section 4.4.

2. Terminology

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

An "application", as used in this document, refers to a candidate protocol, task, or domain to which IPFIX export, collection, and/or storage is applied. By this definition, the IPFIX Applicability statement [RFC5472] defined the initial applications of IPFIX, and PSAMP [RFC5476] was the first new IPFIX application after the publication of the IPFIX protocol itself.

"IANA IE registry", as used in this document, unless otherwise noted, refers to the IANA IPFIX Information Element Registry [iana-ipfix-assignments].

3. How to apply IPFIX

Though originally specified for the export of IP flow information, the message format, template mechanism, and data model specified by IPFIX lead to it being applicable to a wide variety of network management situations. In addition to flow information export, for which it was designed, and packet information export as specified by PSAMP [RFC5476], any application with the following characteristics is a good candidate for an IPFIX application:

- o The application's data flow is fundamentally unidirectional. IPFIX is a "push" protocol, supporting only the export of information from a sender (an Exporting Process) to a receiver (a Collecting Process). Request-response interactions are not supported by IPFIX.
- o The application handles discrete event information, or information to be periodically reported. IPFIX is particularly well suited to representing events, which can be scoped in time.
- o The application handles information about network entities. IPFIX's information model is network-oriented, so network management applications have many opportunities for information model reuse.
- o The application requires a small number of arrangements of data structures relative to the number of records it handles. The template-driven self-description mechanism used by IPFIX excels at handling large volumes of identically structured data, compared to representations which define structure inline with data (such as XML).

Most applications meeting these criteria can be supported over IPFIX. Once it has 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 IANA IE registry may already contain all the necessary Information Elements (see Section 6.1 for guidelines on maximizing Information Element reuse). In this case, no work within the IETF is necessary: simply define Templates and start exporting.

It is expected, however, that most applications will be able to reuse some existing Information Elements, but may need to define some

additional Information Elements to support all their requirements; in this case, see Section 4 for best practices to be followed in defining Information Elements.

Optionally, a Working Group or individual contributor may choose to write an Internet-Draft for publication as an RFC, detailing the new IPFIX application. Such an RFC should contain discussion of the new application, the Information Element definitions as in Section 4, as well as suggested Templates and examples of the use of those Templates within the new application as in Section 9.2. Section 10 defines a compact textual Information Element notation to be used in describing these suggested Templates and/or the use of IPFIX Structured Data [RFC6313] within the new application.

4. Defining new Information Elements

In many cases, a new application will require nothing more than a new Information Element or set of Information Elements to be exportable using IPFIX. An Information Element meeting the following criteria, as evaluated by the IE-DOCTORS, is eligible for inclusion in the IANA IE registry:

- o The Information Element must be unique within the registry, and its description must represent a substantially different meaning from that of any existing Information Element. An existing Information Element that can be reused for a given purpose should be.
- o The Information Element should contain as little internal structure as possible. Instead of representing complex information by overlaying internal structure on a simple data type such as `octetArray`, such information should be represented with multiple simple Information Elements to be exported in parallel or using IPFIX Structured Data [RFC6313], as in Section 4.5. The internal structure of a proposed IE may be evaluated by the IE-DOCTORS with an eye toward interoperability and/or backward compatibility with existing methods of exporting similar data on a case-by-case basis.
- o Information Elements representing information about proprietary or nonstandard applications should not be registered in the IANA IE registry; these be represented using enterprise-specific Information Elements as detailed in section 3.2 [RFC-EDITOR NOTE: verify section number] of [I-D.ietf-ipfix-protocol-rfc5101bis], instead.

The definition of new Information Elements requires a descriptive

name, a specification of the data type from the IPFIX Data Type subregistry in the IANA IE registry (defined in [I-D.ietf-ipfix-information-model-rfc5102bis] as itself extensible via Standards Action as per [RFC5226]), and a human-readable description written in English. This section provides guidelines on each of these components of an Information Element definition, referring to existing documentation such as [I-D.ietf-ipfix-information-model-rfc5102bis] as appropriate.

4.1. Information Element naming

As the name of an Information Element is the first thing a potential implementor will use when determining whether it is suitable for a given application, it is important to be as precise and descriptive as possible. Names of Information Elements:

- o must be chosen carefully to describe the use of the Information Element within the context in which it will be used.
- o must be unique within the IANA IE registry.
- o start with non-capitalized letters.
- o use capital letters for the first letter of each component except for the first one (aka "camel case"). All other letters are non-capitalized, even for acronyms. Exceptions are made for acronyms containing non-capitalized letters, such as 'IPv4' and 'IPv6'. Examples are "sourceMacAddress" and "destinationIPv4Address."

In addition, new Information Elements pertaining to a specific protocol should name the protocol in the first word in order to ease searching by name (e.g. "sipMethod" for a SIP method, as would be used in a logging format for SIP based on IPFIX). Similarly, new Information Elements pertaining to a specific application should name the application in the first word.

4.2. Information Element data types

IPFIX provides a set of data types covering most primitives used in network measurement and management applications. The most appropriate data type should be chosen for the Information Element type, IPFIX informationElementDataTypes subregistry at [iana-ipfix-assignments]. This subregistry may be extended from time to time by a Standards Action [RFC5226], as defined in [RFC5610].

Information Elements representing an integral value with a natural width should be defined with the appropriate integral data type. This applies especially to values taken directly from fixed-width

fields in a measured protocol. For example, `tcpControlBits`, the TCP flags byte, is an `unsigned8`, and `tcpSequenceNumber` is an `unsigned32`.

Information Elements representing counters or identifiers should be defined as `signed64` or `unsigned64`, as appropriate, to maximize the range of values available; applications can use reduced-size encoding as defined in Section 6.2 [RFC-EDITOR NOTE: verify section number] of [I-D.ietf-ipfix-protocol-rfc5101bis] in cases where fewer than 2^{64} values are necessary.

Information Elements representing time values must be defined with appropriate precision. For example, a Information Element for a time measured at second-level precision should be defined as having a `dateTimeSeconds` data type, instead of `dateTimeMilliseconds`.

Information Elements of type `string` or `octetArray` which have length constraints (fixed length, minimum and/or maximum length) must note these constraints in their description.

The type of an Information Element must match the type of the data it represents. More specifically, information that could be represented as a string, but which better matches one of the other data types (e.g. an integral type for a number or enumerated type, an address type for an address) must be represented by the best-matching type, even if the data was represented using a different type in the source. For example, an IPFIX application that exports Options Template Records mapping IP addresses to additional information about each host from an external database must use Information Elements of an address type to represent the addresses, even if the source database represented these as strings.

Strings and `octetArrays` must not be used to encode data that would be more properly represented using multiple Information Elements and/or IPFIX Structured Data [RFC6313]; see Section 4.5 for more.

This document does not cover the addition of new Data Types or Data Type Semantics to the IPFIX Protocol. As such changes have important interoperability considerations and require implementation on both Collecting and Exporting Processes, they require a Standards Action as per [RFC5610]. However, note that the set of primitive types provided by IPFIX are applicable to almost any appropriate application, so extending the type system is generally not necessary.

4.3. Information Element numbering

Each Information Element has a unique identifier in the IANA registry.

When adding newly registered Information Elements to the IANA IE registry, IANA should assign the lowest available Information Element identifier (the value column in [iana-ipfix-assignments]) in the range 128-32767.

Information Elements with identifiers from 1-127 are reserved for compatibility with corresponding fields in NetFlow version 9, as described in [RFC3954].

4.4. Ancillary Information Element properties

Information Elements to which special semantics apply should refer to one of the values in the Information Element Semantics subregistry of the IANA IE registry, as described in Section 3.2 [RFC-EDITOR NOTE: verify section number] of [I-D.ietf-ipfix-information-model-rfc5102bis], subject to the restrictions given in Section 3.10 of [RFC5610]; in other words, the semantics and the type must be consistent.

When defining Information Elements representing a dimensioned quantity or entity count, the units of that quantity should be defined in the units field. This field takes its values from the IANA Information Element Units subregistry of the IANA IE registry. If an Information Element expresses a quantity in units not yet in this subregistry, then the unit must be added to the Units subregistry at the same time the Information Element is added to the IANA IE registry. Note that the Units subregistry as defined in [RFC5610] is maintained on an Expert Review basis.

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's entry the IANA IE registry.

4.5. Internal structure in Information Elements

The definition of Information Elements with internal structure with the structure defined in the Description field is not recommended, except in the following cases:

1. The Information Element is a direct copy of a structured entity in a measured protocol (e.g. the tcpControlBits Information Element for the flags byte from the TCP header)
2. The Information Element represents a section of a packet of protocol entity, in raw form as captured from the wire (e.g. the mplsLabelStackSection Information Element for the MPLS label stack)

3. The Information Element represents a set of flags which are tightly semantically related, where representing the flags as separate one-byte booleans would be inefficient, and which should always appear together in a data record (e.g., the `anonymizationFlags` Information Element for specifying optional features of anonymization techniques)
4. The Information Element contains internal structure by reference to an external datatype or specification containing internal structure (e.g., a MIME type or URL), for interoperability and backward compatibility purposes

Additional exceptions to the above list should be made through a publication of an RFC.

In other cases, candidate Information Elements with internal structure should be decomposed into multiple primitive Information Elements to be used in parallel. For more complicated semantics, where the structure is not identical from Data Record to Data Record, or where there is semantic dependency between multiple decomposed primitive Information Elements, use the IPFIX Structured Data [RFC6313] extension instead.

As an example of information element decomposition, consider an application-level identifier called an "endpoint", which represents a {host, port, protocol} tuple. Instead of allocating an opaque, structured "source endpoint" Information Element, the source endpoint should be represented by three separate Information Elements: "source address", "source port", "transport protocol". In this example, the required information elements already exist in the IANA IE 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 a multitude of templates to handle different multiplicities. More information on multiplicities is given in the following section.

4.6. Information Element multiplicity

Some Information Elements may represent information with a multiplicity other than one; i.e., items that may occur multiple times within the data to be represented in a single IPFIX record. In this case, there are several options, depending on the circumstances:

1. As specified in section 8 [RFC-EDITOR NOTE: verify section number] of [I-D.ietf-ipfix-protocol-rfc5101bis]: "if an Information Element is required more than once in a Template, the different occurrences of this Information Element should follow the logical order of their treatments by the Metering Process." In other words, in cases where the items have a natural order (e.g., the order in which they occur in the packet), and the multiplicity is the same for each record, the information can be modeled by containing multiple instances of the Information Element representing a single item within the Template Record describing the Data Records.
2. In cases where the items have a variable multiplicity, a basicList of the Information Element representing a single item can be used as in the IPFIX Structured Data [RFC6313] extension.
3. If the multiple-item structure is taken directly from bytes observed on the wire by the Metering Process or otherwise taken from the application being measured (e.g., a TCP options stack), the multiple-item structure can be exported as a variable-length octetArray Information Element holding the raw content.

Specifically, new Information Element should not encode any multiplicity or ordinality information into the definition of the Information Element itself.

4.7. Enumerated Values and Subregistries

When defining an Information Element that takes an enumerated value from a set of values which may change in the future, this enumeration must be defined by an IANA IE 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 subregistry of the IANA IPFIX registry must be defined for the enumerated value, to be modified subject to Expert Review [RFC5226].

4.8. Reversibility as per RFC 5103

[RFC5103] defines a method for exporting bidirectional flows using a special Private Enterprise Number to define reverse-direction variants of IANA Information Elements, and a set of criteria for determining whether an Information Element may be reversed using this method. Since almost all Information Elements are reversible, [RFC5103] enumerates those Information Elements which were defined at the time of its publication which are NOT reversible.

New non-reversible Information Elements must contain a note in the

description stating that they are not reversible.

4.9. Avoiding Bad Ideas in Information Element Design

In general, the existence of a similarly-defined Information Element in the IANA IE registry sets a precedent which may be followed to determine whether a given proposed Information Element "fits" within the registry. Indeed, the rules specified by this document could be interpreted to mean "make new Information Elements that look like existing Information Elements". However, for reasons of history, there are several Information Elements within the IANA IE registry which do not follow best practices in Information Element design. These Information Elements are not necessarily so flawed so as to require deprecation, but they should be explicitly ignored when looking for guidance as to whether a new Information Element should be added. Here we provide a set of representative examples taken from the IANA IE registry; in general, entries in the IANA IE registry which do not follow the guidelines in this document should not be used as examples for new Information Element definitions.

Before registering a new Information Element, it must be determined that it would be sufficiently unique within the IANA IE registry. This evaluation has not always been done in the past, and the existence of the Information Elements defined without this evaluation should not be taken as an example that such Information Element definition practices should be followed in the future. Specific examples of such Information Elements include initiatorOctets and responderOctets (which duplicate octetDeltaCount and its reverse per [RFC5103]) and initiatorPackets and responderPackets (the same, for packetDeltaCount).

As mentioned in Section 4.2, the type of an Information Element should match the type of data the Information Element represents. An example of how not to do this is presented by the p2pTechnology, tunnelTechnology, and encryptedTechnology Information Elements: these represent a three-state enumeration using a String. The example set by these Information Elements should not be followed in the definition of new Information Elements.

As mentioned in Section 4.6, an Information Element definition should not include any ordinality or multiplicity information. The only example of this within the IANA IE 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 [RFC6313] where supported, as a basicList of MPLS label entries, or as a raw MPLS label stack using the variable-length `mplsLabelStackSection` Information Element.

5. The Information Element Lifecycle

Once an Information Element or set of Information Elements has been identified for a given application, Information Element specifications in accordance with Section 4 are submitted to IANA to follow the IE-DOCTORS process, as defined below. This process is also used for other changes to the IANA IE registry, such as deprecation or revision, as described later in this section.

5.1. The IE-DOCTORS process

Requests to change the IANA IE registry or a linked subregistry are submitted to IANA, which forwards the request to a designated group of experts (IE-DOCTORS) appointed by the IESG; these are the reviewers called for by the Expert Review [RFC5226] policy defined for the IANA IE registry by [I-D.ietf-ipfix-information-model-rfc5102bis]. The IE-DOCTORS review the request for such things as compliance with this document, compliance with other applicable IPFIX-related RFCs, and consistency with the currently defined set of Information Elements.

Authors are expected to review compliance with the specifications in this document to check their submissions before sending them to IANA.

IE-DOCTORS reviewers should endeavor to complete referred reviews in a timely manner. If the request is acceptable, the IE-DOCTORS signify their approval to IANA, which changes the IANA IE registry. If the request is not acceptable, the IE-DOCTORS can coordinate with the requestor to change the request to be compliant. The IE-DOCTORS may also choose in exceptional circumstances to reject clearly frivolous or inappropriate change requests outright.

This process should not in any way be construed as allowing the IE-DOCTORS to overrule IETF consensus. Specifically, Information Elements in the IANA IE registry which were added with IETF consensus

require IETF consensus for revision or deprecation.

IE-DOCTORS decisions may be appealed as in section 7 of [RFC5226].

5.2. Revising Information Elements

The Information Element status field in the IANA IE registry is defined in [I-D.ietf-ipfix-information-model-rfc5102bis] to allow Information Elements to be 'current' or 'deprecated'. No Information Elements are as of this writing deprecated. [RFC5102] additionally specified an 'obsolete' status; however, this has been removed on revision as it served no operational purpose.

In addition, no policy is defined for revising IANA IE registry entries or addressing errors therein. To be certain, changes and deprecations within the IANA IE 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.

Changes are initiated by sending a new Information Element definition to IANA, as in Section 5.1, for an already-existing Information Element.

The primary requirement in the definition of a policy for managing changes to existing Information Elements is avoidance of interoperability problems; IE-DOCTORS 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:

1. it involves the correction of an error which is obviously only editorial; or
2. it corrects an ambiguity in the Information Element's definition, which itself leads to non-interoperability severe enough to prevent the Information Element's usage as originally defined (e.g., a prior change to `ipv6ExtensionHeaders`); or
3. 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

4. it corrects missing information in the Information Element's definition without changing its meaning (e.g., the explicit definition of 'quantity' semantics for numeric Information Elements without a Data Type Semantics value); or
5. it defines a previously undefined or reserved enumerated value, or one or more previously reserved bits in an Information Element with flag semantics; or
6. it expands the set of permissible values in the Information Element's range; or
7. it harmonizes with an external reference which was itself corrected.

If a change is deemed permissible by the IE-DOCTORS, IANA makes the change in the IANA IE registry. The requestor of the change is appended to the Requestor in the registry.

Each Information Element in the IANA IE registry has a Revision number, starting at zero. Each change to an Information Element following this process increments the Revision number by one. Since any revision must be interoperable according to the criteria above, there is no need for the IANA IE registry to store information about old revisions.

When a revised Information Element is accepted into the registry, the date of acceptance of the most recent revision is placed into the revision Date column of the registry for that Information Element.

5.3. Deprecating Information Elements

Changes that are not permissible by these criteria may only be handled by deprecation. An Information Element MAY be deprecated and replaced when:

1. the Information Element definition has an error or shortcoming which cannot be permissibly changed as in Section 5.2; or
2. the deprecation harmonizes with an external reference which was itself deprecated through that reference's accepted deprecation method; or
3. 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 requires a Standards Action.

A request for deprecation is sent to IANA, which passes it to the IE-DOCTORS for review, as in Section 5.1. When deprecating an Information Element, the Information Element description in the IANA IE registry must be updated to explain the deprecation, as well as to refer to any new Information Elements created to replace the deprecated Information Element.

The Revision number of an Information Element is incremented upon deprecation, and the revision Date updated, as with any revision.

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.

Names and elementIDs of deprecated Information Elements must not be reused.

6. When not to define new Information Elements

Due to the relatively limited number space of Information Elements in the IANA IE registry, and the fact that the difficulty of managing and understanding the registry increases with its size, avoiding redundancy and clutter in the registry is important in defining new applications. New Information Elements should not be added to the IANA IE registry unless there is an intent to implement and deploy applications using them; research or experimental applications should use enterprise-specific Information Elements as in Section 6.2 instead.

The subsections below provide guidelines for reuse of existing Information Elements, as well as guidelines on using enterprise-specific Information Elements instead of adding Information Elements in the IANA IE 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 should ensure that the existing one is not applicable.

Note that this guideline to maximize reuse does not imply that an Information Element that represents the same information from a packet as a existing Information Element should not be added to the IANA IE registry. For example, consider the `ipClassOfService` (Element ID 5), `ipDiffServCodePoint` (Element ID 98), and `ipPrecedence` (Element ID 196) Information Elements. These all represent subsets of the same field in an IP version 4 packet header, but different uses of these bits. The representation in one or another of these Information Elements contains information in itself as to how the bits were interpreted by the Metering Process.

On the other hand, simply changing the context in which an Information Element will be used is insufficient reason for the definition of a new Information Element. For example, an extension of IPFIX to log detailed information about HTTP transactions alongside network-level information should not define `httpClientAddress` and `httpServerAddress` Information Elements, preferring instead the use of `sourceIPv[46]Address` and `destinationIPv[46]Address`.

Applications dealing with bidirectional interactions should use Bidirectional Flow Support for IPFIX [RFC5103] to represent these interactions.

Existing timestamp and time range Information Elements should be reused for any situation requiring simple time stamping of an event: for single observations, the `observationTime*` Information Elements from PSAMP are provided, and for events with a duration, the `flowStart*` and `flowEnd*` Information Elements suffice. This arrangement allows minimal generic time handling by existing Collecting Processes and analysis workflows. New timestamp Information Elements should ONLY be defined for semantically distinct timing information (e.g., an IPFIX-exported record containing information about an event to be scheduled in the future).

In all cases the use of absolute timestamp Information Elements (e.g. `flowStartMilliseconds`) is recommended, as these Information Elements allow for maximum flexibility in processing with minimal overhead. Timestamps based on the export time header in the enclosing IPFIX Message (e.g. `flowStartTimeDeltaMicroseconds`) MAY be used if high-precision timing is important, export bandwidth or storage space is limited, timestamps comprise a relatively large fraction of record size, and the application naturally groups records into IPFIX Messages. Timestamps based on information which must be exported in a separate Data Record defined by an Options Template (e.g. `flowStartSysUpTime`) MAY be used only in the context of an existing practice of using runtime-defined epochs for the given application. New applications should avoid these structures when possible.

6.2. Applying enterprise-specific Information Elements

IPFIX provides a mechanism for defining enterprise-specific Information Elements, as in Section 3.2 [RFC-EDITOR NOTE: verify section number] of [I-D.ietf-ipfix-protocol-rfc5101bis]. These are scoped to a vendor's or organization's Structure of Management Information (SMI) Private Enterprise Number, and are under complete control of the organization assigning them.

For situations in which interoperability is unimportant, new information should be exported using enterprise-specific Information Elements instead of adding new Information Elements to the IANA IE registry. These situations include:

- o export of implementation-specific information, or
- o export of information supporting research or experiments within a single organization or closed community, 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 IANA IE registry. Instead, these experimental Information Elements should be represented as enterprise-specific until their definitions are finalized.

As this document contains best practices for defining new Information Elements, organizations using enterprise-specific Information Elements are advised to follow the guidelines set forth here even if not submitting Information Elements for inclusion in the IANA IE registry.

7. Information Element Definition Checklist

The following three checklists, condensed from the rest of this document, can be used when defining and reviewing Information Elements; they refer back to the section of this document from which they are taken. These checklists are intended for the definition of

new Information Elements; revision should follow the process defined in Section 5.2, and deprecation should follow the process defined in Section 5.3.

Though many of the considerations in this document require the subjective judgement of Information Element authors, reviewers, and IANA, certain parts of the process may be made simpler through tool support. Items on these checklists which could be easily automated or assisted by tools are annotated with "(tool support)". Other items on these checklists require some level of subjective judgement; checks for semantic uniqueness may additionally be supported by textual analysis of descriptions in the future.

Checklist 1 contains conditions which must be met by all proposed Information Elements:

1. The name must be unique within the IANA IE registry, and not reuse the name of any current or deprecated Information Element. (Section 4.1) (tool support)
2. The description must be sufficiently semantically unique within the IANA IE registry, representing a substantially different meaning from any current or deprecated Information Element. (Section 4)
3. The name must start with a non-capitalized letter. (Section 4.1) (tool support)
4. Names composed of more than one word must use capital letters for the first letter of each component except for the first one; all other letters are non-capitalized, even for acronyms. Exceptions are made for acronyms containing non-capitalized letters, such as 'IPv4' and 'IPv6'. (Section 4.1) (tool support)
5. The data type must match the type of the data being represented. (Section 4.2)
6. Data type semantics must be appropriate for the data type. (Section 4.4) (tool support)
7. The Information Element identifier assigned by IANA must be unique. (Section 4.3) (tool support)
8. The Information Element must be reviewed for the potential of information leakage or other misuse that could reduce the security of the measured system; security considerations specific to the Information Element must be discussed in the description or in a supporting RFC. (Section 11)

Checklist 2 contains conditions which must be met by proposed Information Elements with certain properties, as noted:

1. Time values must be defined with appropriate precision. (Section 4.2)
2. Strings and octet arrays with length restrictions must note those length restrictions in their descriptions. (Section 4.2)
3. Enumerations must refer to an IANA IE registry or subregistry, or a registry maintained by an external standards organization. If no suitable registry or subregistry exists, a new subregistry of the IPFIX Information Element registry must be created for the enumeration, to be modified subject to Expert Review [RFC5226]. (Section 4.7)

Checklist 3 contains conditions which should be met by proposed Information Elements:

1. The name of an Information Element pertaining to a specific protocol or application should contain the name of the protocol or application as the first word. (Section 4.1)
2. Information Elements representing integral values should use a data type for the appropriate width for the value. (Section 4.2)
3. Information Elements representing counters or identifiers should be represented as signed64 or unsigned64, unless they are naturally represented with narrower integral types, as appropriate. (Section 4.2)
4. An Information Element should not contain internal structure, subject to the exceptions in Section 4.5; candidate Information Elements with internal structure should be decomposed into multiple Information Elements. (Section 4.5)
5. An Information Element should not contain multiplicity or ordinality information within the definition of the Information Element itself. (Section 4.6)
6. Data type semantics should be defined, if appropriate. (Section 4.4) (tool support)
7. Units should be defined, if appropriate, with new units added to the Information Element Units subregistry if necessary. (Section 4.4) (tool support)

8. Ranges should be defined, if appropriate. (Section 4.4) (tool support)
9. Non-reversible Information Elements (see [RFC5103]) should note non-reversibility in the description. (Section 4.8)
10. Information Elements to be registered with IANA should be intended for implementation and deployment on production networks.

8. Applying IPFIX to non-Flow Applications

At the core of IPFIX is its definition of a Flow, a set of packets sharing some common properties crossing an Observation Point within a certain time window. However, the reliance on this definition does not preclude the application of IPFIX to domains which are not obviously handling flow data according to this definition. Most network management data collection tasks, those to which IPFIX is most applicable, have at their core the movement of packets from one place to another; by a liberal interpretation of the common properties defining the flow, then, almost any event handled by these can be held to concern data records conforming to the IPFIX definition of a Flow.

Non-flow information defining associations or key-value pairs, on the other hand, are defined by IPFIX Options Templates. Here, the Information Elements within an Options Template Record are divided into Scope Information Elements which define the key, and non-scope Information Elements which define the values associated with that key. Unlike Flows, Data Records defined by Options Template are not necessarily scoped in time; these Data Records are generally held to be in effect until a new set of values for a specific set of keys is exported. While this mechanism is often used by IPFIX to export metadata about the collection infrastructure, it is applicable to any association information.

An IPFIX application can mix Data Records described either type of template in an IPFIX Message or Message stream, and exploit relationships among the Flow Keys, values, and Scopes to create interrelated data structures. See [RFC5473] for an example application of this.

9. Writing Internet-Drafts for IPFIX Applications

When a new application is complex enough to require additional clarification or specification as to the use of the defined

Information Elements, or has particularly this may be given in an Internet-Draft. Internet-Drafts for new IPFIX applications are best submitted to a Working Group with expertise in the area of the new application, or as independent submissions.

When defining new Information Elements in an Internet-Draft, the Internet-Draft should contain a section (or subsection) for each Information Element, which contains the attributes in Section 4 in human-readable form. An example subsection is given below. These Information Element descriptions should not assign Information Element numbers, instead using placeholder identifiers for these numbers (e.g. "TBD1", "TBD2", "TBD3") and a note to IANA in the IANA Considerations section to replace those placeholders in the document with Information Element numbers when the numbers are assigned. The use of these placeholder definitions allows references to the numbers in e.g. box-and-line diagrams or template definitions as in Section 10.

9.1. Example Information Element Definition

This is an example of an Information Element definition which would appear in an Internet-Draft. The name appears in the section title.

Description: Description goes here.; obligatory

Data Type: Data type goes here; obligatory

Data Type Semantics: Data type semantics, if any, go here; optional

Units: Units, if any, go here; optional

Range: Range, if not implied by the data type, goes here; optional

References: References to other RFCs or documents outside the IETF, in which additional information is given, or which are referenced by the description, go here; optional

ElementId: ElementId, if known, or TBD to be filled in by IANA at publication time.

9.2. Defining Recommended Templates

New IPFIX applications should not, in the general case, define fixed templates for export, as this throws away much of the flexibility afforded by IPFIX. However, fixed template export is permissible in the case that the export implementation must operate in a resource constrained environment, and/or that the application is replacing an existing fixed-format binary export format in a maximally compatible

way. In any case, Collecting Processes for such applications should support the collection Templates with Information Elements in any order, or Templates with additional Information Elements.

An Internet-Draft clarifying the use of new Information Elements should include any recommended Template or Options Template Records necessary for supporting the application, as well as examples of records exported using these Template Records. In defining these Template Records, such Internet-Drafts should mention, subject to rare exceptions:

1. that the order of different Information Elements within a Template is not significant;
2. that Templates on the wire for the application may also contain additional Information Elements beyond those specified in the recommended Template;
3. that a stream of IPFIX Messages supporting the application may also contain Data Records not described by the recommended Templates; and
4. that any reader of IPFIX Messages supporting the application must accept these conditions.

Definitions of recommended Template Records for flow-like information, where the Flow Key is well-defined, should indicate which of the Information Elements in the recommended Template are Flow Keys.

Recommended Templates are defined, for example, in [RFC5476] for PSAMP packet reports (section 6.4) and extended packet reports (section 6.5). Recommended Options Templates are defined extensively throughout the IPFIX documents, including in the protocol document itself [I-D.ietf-ipfix-protocol-rfc5101bis] for exporting export statistics; in the file format [RFC5655] for exporting file metadata; and in Mediator intermediate process definitions such as [RFC6235] for intermediate process metadata. The discussion in these examples is a good model for recommended template definitions.

10. A Textual Format for Specifying Information Elements and Templates

Example Templates given in existing IPFIX documents are generally expressed using bitmap diagrams of the respective Templates. These are illustrative of the wire representation of simple Templates, but not particularly readable for more complicated recommended Templates, provide no support for rapid implementation of new Templates, and do

not adequately convey the optional nature of ordering and additional Information Elements. Therefore, we define a recommended textual format for specifying Information Elements and Templates in Internet-Drafts in this section.

Here we define a simple textual syntax for describing IPFIX Information Elements and IPFIX Templates, with human readability, human writability, compactness, and ease of parser/generator implementation without requiring external XML support as design goals. It is intended both for use in human communication (e.g., in new Internet-Drafts containing higher-level descriptions of IPFIX Templates, or describing sets of new IPFIX Information Elements for supporting new applications of the protocol) as well as at runtime by IPFIX implementations.

10.1. Information Element Specifiers

The basis of this format is the textual Information Element Specifier, or IESpec. An IESpec contains each of the four important aspects of an Information Element: its name, its number, its type, and its size, separated by simple markup based on various types of brackets. Fully-qualified IESpecs may be used to specify existing or new Information Elements within an Information Model, while either fully-qualified or partial IESpecs may be used to define fields in a Template.

Bare words are used for Information Element names, and each aspect of information associated with an Information Element is associated with a type of brackets:

- o () parentheses for Information Element numbers,
- o <> angles for Information Element data types, and
- o [] square brackets for Information Element sizes.
- o { } curly braces contain an optional space-separated list of context identifiers to be associated with an Information Element, as described in more detail in Section 10.2

The symbol + is reserved for Information Element nesting within structured data elements; these are described in Section 10.3.

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 in the IANA IE registry. At minimum, a partial IESpec must contain a name, or a number. Any name, number, or type information given with a partial IESpec must match the values given in the Information Model; however, size information in a partial IESpec overrides size information in the Information Model; in this way, IESpecs can be used to express reduced-length encoding for Information Elements.

Example partial IESpecs follow:

- o `octetDeltaCount`
- o `octetDeltaCount[4]` (reduced-length encoding)
- o (1)
- o (1)[4] (reduced length encoding; note that this is exactly equivalent to an Information Element specifier in a Template)

10.2. Specifying Templates

A Template can then be defined simply as an ordered, newline-separated sequence of IESpecs. IESpecs in example Templates illustrating a new application of IPFIX should be fully-qualified. Flow Keys may be optionally annotated by appending the {key} context to the end of each Flow Key specifier. A template counting packets and octets per five-tuple with millisecond precision in IESpec syntax is shown below.

```
flowStartMilliseconds(152)<dateTimeMilliseconds>[8]
flowEndMilliseconds(153)<dateTimeMilliseconds>[8]
octetDeltaCount(1)<unsigned64>[8]
packetDeltaCount(2)<unsigned64>[8]
sourceIPv4Address(8)<ipv4Address>[4]{key}
destinationIPv4Address(12)<ipv4Address>[4]{key}
sourceTransportPort(7)<unsigned16>[2]{key}
destinationTransportPort(11)<unsigned16>[2]{key}
protocolIdentifier(4)<unsigned8>[1]{key}
```

Fig. 1: Sample flow template in IESpec syntax

An Options Template is specified similarly. Scope is specified appending the {scope} context to the end of each IESpec for a Scope IE. Due to the way Information Elements are represented in Options Templates, all {scope} IESpecs must appear before any non-scope IESpec. The Flow Key Options Template defined in section 4.4 [RFC-EDITOR NOTE: verify section number] of [I-D.ietf-ipfix-protocol-rfc5101bis] in IESpec syntax is shown below:

```
templateId(145)<unsigned16>[2]{scope}
flowKeyIndicator(173)<unsigned64>[8]
```

Fig. 2: Flow Key Options Template in IESpec syntax

10.3. Specifying IPFIX Structured Data

IESpecs can also be used to illustrate the structure of the information exported using the IPFIX Structured Data extension [RFC6313]. Here, the semantics of the structured data elements are specified using contexts, and the information elements within each structured data element follow the structured data element, prefixed with + to show they are contained therein. Arbitrary nesting of structured data elements is possible by using multiple + signs in the prefix. For example, a basic list of IP addresses with "one or more" semantics would be expressed using partially qualified IESpecs as follows:

```
basicList{oneOrMoreOf}  
+sourceIPv4Address(8)[4]
```

Fig. 3: Sample basicList in IESpec syntax

And an example subTemplateList itself containing a basicList is shown below:

```
subTemplateList{allOf}  
+basicList{oneOrMoreOf}  
++sourceIPv4Address(8)[4]  
+destinationIPv4Address(12)[4]
```

Fig. 4: Sample subTemplateList in IESpec syntax

This describes a subTemplateMultilist containing all of the expressed set of source-destination pairs, where the source address itself could be one of any number in a basicList (e.g., in the case of SCTP multihoming).

The contexts associable with structured data Information Elements are the semantics, as defined in section 4.4 of [RFC6313]; a structured data Information Element without any context is taken to have undefined semantics. More information on the application of structured data is available in [RFC6313].

11. Security Considerations

The IE-DOCTORS must evaluate the security aspects of new Information Elements in light of the information they could provide to support potential attacks against the measured network or entities about which information is exported. Specific security aspects to evaluate include whether the exported information contains personally identifiable information, or information which should be kept

confidential about the described entities (e.g. partial payload, or configuration information which could be exploited). This is not to say that such Information Elements should not be defined, but there must be an evaluation of the security risk versus the utility of the exported information for the intended application. For example, "A Framework for Packet Selection and Reporting" [RFC5474] concluded in section 12.3.2 that the hash functions private parameters should not be exported within IPFIX.

Security considerations specific to an Information Element must be addressed in the Security Considerations section of the Internet-Draft describing the Information Element, or in the Information Element description itself in case the Information Element is not defined in an Internet-Draft. Information Elements with specific security considerations should be described in an Internet-Draft.

For example, the `ipHeaderPacketSection` in the IPFIX IE 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." Another example can be seen in 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."

12. IANA Considerations

This document has no considerations for IANA.

[IANA NOTE: IANA considerations for the process outlined in this document are given in [I-D.ietf-ipfix-information-model-rfc5102bis]]

[IANA NOTE: Note that the examples in Appendix A are NOT to be added to the IPFIX Information Element registry upon the publication of this document.]

13. Acknowledgements

Thanks to Paul Aitken, Andrew Feren, Dan Romascanu, and David Harrington for their reviews and feedback. Thanks as well to Roni

Even and Yoav Nir for their area reviews; and to Pete Resnick, Adrian Farrel, Stephen Farrell, Stewart Bryant, and Barry Leiba for their contributions during IESG discussions. This work is materially supported by the European Union Seventh Framework Programme under grant agreement 257315 (DEMONS).

14. References

14.1. Normative References

- [I-D.ietf-ipfix-protocol-rfc5101bis]
Claise, B. and B. Trammell, "Specification of the IP Flow Information eXport (IPFIX) Protocol for the Exchange of Flow Information", draft-ietf-ipfix-protocol-rfc5101bis-02 (work in progress), June 2012.
- [I-D.ietf-ipfix-information-model-rfc5102bis]
Claise, B. and B. Trammell, "Information Model for IP Flow Information eXport (IPFIX)", draft-ietf-ipfix-information-model-rfc5102bis-05 (work in progress), September 2012.
- [RFC5103] Trammell, B. and E. Boschi, "Bidirectional Flow Export Using IP Flow Information Export (IPFIX)", RFC 5103, January 2008.
- [RFC5610] Boschi, E., Trammell, B., Mark, L., and T. Zseby, "Exporting Type Information for IP Flow Information Export (IPFIX) Information Elements", RFC 5610, July 2009.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.
- [RFC6313] Claise, B., Dhandapani, G., Aitken, P., and S. Yates, "Export of Structured Data in IP Flow Information Export (IPFIX)", RFC 6313, July 2011.

14.2. Informative References

- [RFC2804] IAB and IESG, "IETF Policy on Wiretapping", RFC 2804, May 2000.
- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

- [RFC3954] Claise, B., "Cisco Systems NetFlow Services Export Version 9", RFC 3954, October 2004.
- [RFC5102] Quittek, J., Bryant, S., Claise, B., Aitken, P., and J. Meyer, "Information Model for IP Flow Information Export", RFC 5102, January 2008.
- [RFC5472] Zseby, T., Boschi, E., Brownlee, N., and B. Claise, "IP Flow Information Export (IPFIX) Applicability", RFC 5472, March 2009.
- [RFC5473] Boschi, E., Mark, L., and B. Claise, "Reducing Redundancy in IP Flow Information Export (IPFIX) and Packet Sampling (PSAMP) Reports", RFC 5473, March 2009.
- [RFC5474] Duffield, N., Chiou, D., Claise, B., Greenberg, A., Grossglauser, M., and J. Rexford, "A Framework for Packet Selection and Reporting", RFC 5474, March 2009.
- [RFC5476] Claise, B., Johnson, A., and J. Quittek, "Packet Sampling (PSAMP) Protocol Specifications", RFC 5476, March 2009.
- [RFC5477] Dietz, T., Claise, B., Aitken, P., Dressler, F., and G. Carle, "Information Model for Packet Sampling Exports", RFC 5477, March 2009.
- [RFC5560] Uijterwaal, H., "A One-Way Packet Duplication Metric", RFC 5560, May 2009.
- [RFC5655] Trammell, B., Boschi, E., Mark, L., Zseby, T., and A. Wagner, "Specification of the IP Flow Information Export (IPFIX) File Format", RFC 5655, October 2009.
- [RFC6235] Boschi, E. and B. Trammell, "IP Flow Anonymization Support", RFC 6235, May 2011.
- [iana-ipfix-assignments]
Internet Assigned Numbers Authority, "IP Flow Information Export Information Elements
(<http://www.iana.org/assignments/ipfix/ipfix.xml>)".

Appendix A. Example Information Element Definitions

This section contains a few example Information Element definitions as they would appear in an Internet-Draft. Note the conformance of these examples to the guidelines in Section 4.

The sipResponseStatus Information Element (Appendix A.1) illustrates the addition of an Information Element representing Layer 7 application information, with a reference to the registry containing the allowable values. The duplicatePacketDeltaCount Information Element (Appendix A.2) illustrates the addition of a new metric, with a reference to the RFC defining the metric. The ambientTemperature Information Element (Appendix A.3) illustrates the addition of a new measured value outside the area of traditional networking applications.

A.1. sipResponseStatus

Description: The SIP Response code as an integer, as in the Response Codes registry at <http://www.iana.org/assignments/sip-parameters> defined in [RFC3261] and amended in subsequent RFCs. The presence of this Information Element in a SIP Message record marks it as describing a SIP response; if absent, the record describes a SIP request.

Data Type: unsigned16

Data Type Semantics: identifier

References: [RFC3261]

ElementId: TBD

Replaces Enterprise-Specific Element: 35566 / 412

A.2. duplicatePacketDeltaCount

Description: The number of uncorrupted and identical additional copies of each individual packet in the Flow arriving at the destination since the previous Data Record for this Flow (if any), as measured at the Observation Point. This is measured as the Type-P-one-way-packet-duplication metric defined in section 3 of [RFC5560].

Data Type: unsigned64

Data Type Semantics: deltaCounter

Units: packets

References: [RFC5560]

ElementId: TBD

A.3. ambientTemperature

Description: An ambient temperature observed by measurement equipment at an Observation Point, positioned such that it measures the temperature of the surroundings (i.e., not including any heat generated by the measuring or measured equipment), expressed in degrees Celsius.

Data Type: float

Units: degrees Celsius

Range: -273.15 - +inf

ElementId: TBD

Authors' Addresses

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

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

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

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

Network Working Group
Internet Draft
Obsoletes: 5102
Category: Standards Track
Expires: August 16, 2013

B. Claise, Ed.
Cisco Systems, Inc.
B. Trammell, Ed.
ETH Zurich
February 12, 2013

Information Model for IP Flow Information eXport (IPFIX)
draft-ietf-ipfix-information-model-rfc5102bis-10.txt

Abstract

This document defines the datatypes and management policy for the information model for the IP Flow Information eXport (IPFIX) protocol. This information model is maintained as the IANA IPFIX Information Element Registry, the initial contents of which were defined by RFC 5102. This information model is used by the IPFIX Protocol for encoding measured traffic information and information related to the traffic Observation Point, the traffic Metering Process, and the Exporting Process. Although developed for the IPFIX Protocol, the model is defined in an open way that easily allows using it in other protocols, interfaces, and applications. This document obsoletes RFC 5102.

Status of This Memo

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

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

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

This Internet-Draft will expire on August 16, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

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

Table of Contents

1. Introduction	3
1.1. Changes since RFC 5102	4
1.2. IPFIX Documents Overview	4
2. Properties of IPFIX Protocol Information Elements	5
2.1. Information Element Specification Template	5
2.2. Scope of Information Elements	7
2.3. Naming Conventions for Information Elements	7
3. Type Space	8
3.1. Abstract Data Types	8
3.1.1. unsigned8	9
3.1.2. unsigned16	9
3.1.3. unsigned32	9
3.1.4. unsigned64	9
3.1.5. signed8	9
3.1.6. signed16	9
3.1.7. signed32	9
3.1.8. signed64	9
3.1.9. float32	10
3.1.10. float64	10
3.1.11. boolean	10
3.1.12. macAddress	10
3.1.13. octetArray	10
3.1.14. string	10
3.1.15. dateTimeSeconds	10
3.1.16. dateTimeMilliseconds	10
3.1.17. dateTimeMicroseconds	10
3.1.18. dateTimeNanoseconds	11
3.1.19. ipv4Address	11
3.1.20. ipv6Address	11
3.1.21. basicList	11
3.1.22. subTemplateList	11
3.1.23. subTemplateMultiList	11
3.2. Data Type Semantics	11
3.2.1. quantity	11
3.2.2. totalCounter	12
3.2.3. deltaCounter	12
3.2.4. identifier	12

3.2.5. flags	12
4. Information Element Identifiers	13
5. Information Elements	13
6. Extending the Information Model	15
7. IANA Considerations	15
7.1. IPFIX Information Elements	16
7.2. MPLS Label Type Identifier	16
7.3. XML Namespace and Schema	17
7.4. Addition, Revision, and Deprecation	17
8. Security Considerations	18
9. Acknowledgements	19
10. References	19
10.1. Normative References	19
10.2. Informative References	19
Authors' Addresses	22
Contributors' Addresses	22

1. Introduction

The IP Flow Information eXport (IPFIX) protocol serves for transmitting information related to network traffic measurement. The protocol specification in [RFC5101bis] defines how Information Elements are transmitted. For Information Elements, it specifies the encoding of a set of basic data types. However, the list of Information Elements that can be transmitted by the protocol, such as Flow attributes (source IP address, number of packets, etc.) and information about the Metering and Exporting Process (packet Observation Point, sampling rate, Flow timeout interval, etc.), is not specified in [RFC5101bis].

The IANA IPFIX Information Element registry [IPFIX-IANA] is the current complete reference for IPFIX Information Elements. The initial values for this registry were provided by [RFC5102].

This document complements the IPFIX protocol specification [RFC5101bis] by providing an overview of the IPFIX information model and specifying data types for it. IPFIX-specific terminology used in this document is defined in Section 2 of [RFC5101bis]. As in [RFC5101bis], these IPFIX-specific terms have the first letter of a word capitalized when used in this document.

The use of the term 'information model' is not fully in line with the definition of this term in [RFC3444], as the IPFIX information model does not specify relationships between Information Elements. Nor does the IPFIX information model specify a concrete encoding of Information Elements; for an encoding suitable for use with the IPFIX protocol, see [RFC5101bis]. Besides the encoding used by the IPFIX

protocol, other encodings of IPFIX Information Elements can be applied, for example, XML-based encodings.

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

1.1. Changes since RFC 5102

This document obsoletes the Proposed Standard revision of the IPFIX Protocol Specification [RFC5102]. The following changes have been made to this document with respect to the previous document:

- All outstanding technical and editorial errata filed on the [RFC5102] as of publication time have been corrected.
- All references into [RFC5101] have been updated to [RFC5101bis], reflecting changes in that document as necessary.
- Information element definitions have been removed, as the reference for these is now [IPFIX-IANA]; a historical note on categorizations of information elements as defined in [RFC5102] has been retained in section 5.
- The process for modifying [IPFIX-IANA] has been improved, and is now described in [IPFIX-IE-DOCTORS]; Section 6 has been updated accordingly, and a new section 7.3 gives IANA considerations for this process.
- Definitions of timestamp data types have been clarified.
- Appendices A and B have been removed

1.2. IPFIX Documents Overview

The IPFIX protocol provides network administrators with access to network flow information. The architecture for the export of measured flow information out of an IPFIX Exporting Process to a Collecting Process is defined in [RFC5470], per the requirements defined in [RFC3917]. The IPFIX Protocol Specification [RFC5101bis] defines how IPFIX data records and templates are carried via a number of transport protocols from IPFIX Exporting Processes to IPFIX Collecting Processes.

Four IPFIX optimizations/extensions are currently specified: a bandwidth saving method for the IPFIX protocol in [RFC5473], an efficient method for exporting bidirectional flows in [RFC5103], a method for the definition and export of complex data structures in [RFC6313], and the specification of the Protocol for IPFIX Mediations [IPFIX-MED-PROTO] based on the IPFIX Mediation Framework [RFC6183].

IPFIX has a formal description of IPFIX Information Elements, their name, type and additional semantic information, as specified in this document, with the export of the Information Element types specified in [RFC5610].

[RFC6728] specifies a data model for configuring and monitoring IPFIX and PSAMP compliant devices using the NETCONF protocol, while [RFC6615] specifies a MIB module for monitoring.

In terms of development, [RFC5153] provides guidelines for the implementation and use of the IPFIX protocol, while [RFC5471] provides guidelines for testing.

Finally, [RFC5472] describes what type of applications can use the IPFIX protocol and how they can use the information provided. It furthermore shows how the IPFIX framework relates to other architectures and frameworks.

2. Properties of IPFIX Protocol Information Elements

2.1. Information Element Specification Template

Information in messages of the IPFIX protocol is modeled in terms of Information Elements of the IPFIX information model. The IPFIX Information Elements mentioned in Section 5 are specified in [IPFIX-IANA].

All Information Elements specified for the IPFIX protocol MUST have the following properties defined.

name - A unique and meaningful name for the Information Element.

elementId - A numeric identifier of the Information Element. If this identifier is used without an enterprise identifier (see [RFC5101bis] and enterpriseId below), then it is globally unique and the list of allowed values is administered by IANA. It is used for compact identification of an Information Element when encoding Templates in the protocol.

description - The semantics of this Information Element. Describes how this Information Element is derived from the Flow or other information available to the observer. Information Elements of dataType string or octetArray which have length constraints (fixed length, minimum and/or maximum length) MUST note these constraints in their description.

dataType - One of the types listed in Section 3.1 of this document or registered in the IANA IPFIX Information Element Data Types registry. The type space for attributes is constrained to facilitate implementation. The existing type space encompasses most primitive types used in modern programming languages, as well as some derived types (such as `ipv4Address`) that are common to this domain.

status - The status of the specification of this Information Element. Allowed values are 'current' and 'deprecated'. All newly-defined Information Elements have 'current' status. The process for moving Information Elements to the 'deprecated' status is defined in Section 5.2 of [IPFIX-IE-DOCTORS].

Enterprise-specific Information Elements MUST have the following property defined:

enterpriseId - Enterprises may wish to define Information Elements without registering them with IANA, for example, for enterprise-internal purposes. For such Information Elements, the Information Element identifier described above is not sufficient when the Information Element is used outside the enterprise. If specifications of enterprise-specific Information Elements are made public and/or if enterprise-specific identifiers are used by the IPFIX protocol outside the enterprise, then the enterprise-specific identifier MUST be made globally unique by combining it with an enterprise identifier. Valid values for the **enterpriseId** are defined by IANA as Structure of Management Information (SMI) network management private enterprise numbers, defined at [PEN-IANA].

All Information Elements specified for the IPFIX protocol either in this document or by any future extension MAY have the following properties defined:

dataTypeSemantics - The integral types are qualified by additional semantic details. Valid values for the data type semantics are specified in Section 3.2 of this document or in a future extension of the information model.

units - If the Information Element is a measure of some kind, the units identify what the measure is.

range - Some Information Elements may only be able to take on a restricted set of values that can be expressed as a range (e.g., 0 through 511 inclusive). If this is the case, the valid inclusive range SHOULD be specified; values for this Information Element outside the range are invalid and MUST NOT be exported.

reference - Identifies additional specifications that more precisely define this item or provide additional context for its use.

The following two Information Element properties are defined to allow the management of an Information Element registry with Information Element definitions that may be updated over time, per the process defined in Section 5.2 of [IPFIX-IE-DOCTORS].

revision - The revision number of an Information Element, starting at 0 for Information Elements at time of definition, and incremented by one for each revision.

date - The date of the entry of this revision of the Information Element into the registry.

A template for specifying Information Elements in Internet-Drafts is given in Section 9.1 of [IPFIX-IE-DOCTORS], and an XML Schema for specifying Information Elements in the IANA IPFIX registry [IPFIX-IANA] at [IPFIX-XML-SCHEMA].

2.2. Scope of Information Elements

By default, most Information Elements have a scope specified in their definitions. Within Data Records defined by Option Templates, the IPFIX protocol allows further limiting of the Information Element scope. The new scope is specified by one or more scope fields and defined as the combination of all specified scope values; see Section 3.4.2.1 on IPFIX scopes in [RFC5101bis].

2.3. Naming Conventions for Information Elements

The following naming conventions were used for naming Information Elements in this document. It is recommended that extensions of the model use the same conventions.

- o Names of Information Elements SHOULD be descriptive.
- o Names of Information Elements MUST be unique within the IANA IPFIX registry [IPFIX-IANA]. Enterprise-specific Information Elements SHOULD be prefixed with a vendor name.
- o Names of Information Elements MUST start with non-capitalized letters.

- o Composed names MUST use capital letters for the first letter of each component (except for the first one). All other letters are non-capitalized, even for acronyms. Exceptions are made for acronyms containing non-capitalized letters, such as 'IPv4' and 'IPv6'. Examples are `sourceMacAddress` and `destinationIPv4Address`.
- o Middleboxes [RFC3234] may change Flow properties, such as the Differentiated Service Code Point (DSCP) value or the source IP address. If an IPFIX Observation Point is located in the path of a Flow before one or more middleboxes that potentially modify packets of the Flow, then it may be desirable to also report Flow properties after the modification performed by the middleboxes. An example is an Observation Point before a packet marker changing a packet's IPv4 Type of Service (TOS) field that is encoded in Information Element `ipClassOfService`. Then the value observed and reported by Information Element `ipClassOfService` is valid at the Observation Point, but not after the packet passed the packet marker. For reporting the change value of the TOS field, the IPFIX information model uses Information Elements that have a name prefix "post", for example, "postIpClassOfService". Information Elements with prefix "post" report on Flow properties that are not necessarily observed at the Observation Point, but which are obtained within the Flow's Observation Domain by other means considered to be sufficiently reliable, for example, by analyzing the packet marker's marking tables.

3. Type Space

This section describes the abstract data types that can be used for the specification of IPFIX Information Elements in Section 4. Section 3.1 describes the set of abstract data types.

Abstract data types `unsigned8`, `unsigned16`, `unsigned32`, `unsigned64`, `signed8`, `signed16`, `signed32`, and `signed64` are integral data types. As described in Section 3.2, their data type semantics can be further specified, for example, by `'totalCounter'`, `'deltaCounter'`, `'identifier'`, or `'flags'`.

3.1. Abstract Data Types

This section describes the set of valid abstract data types of the IPFIX information model, independent of encoding. Note that further abstract data types may be specified by future updates to this document. Changes to the associated IPFIX Information Element Data Types subregistry [IPFIX-IANA] specified in [RFC5610] require a Standards Action [RFC5226].

The current encodings of these data types for use with the IPFIX protocol is defined in [RFC5101bis]; encodings allowing the use of the IPFIX Information Elements [IPFIX-IANA] with other protocols may be defined in the future by referencing this document.

3.1.1. unsigned8

The type "unsigned8" represents a non-negative integer value in the range of 0 to 255.

3.1.2. unsigned16

The type "unsigned16" represents a non-negative integer value in the range of 0 to 65535.

3.1.3. unsigned32

The type "unsigned32" represents a non-negative integer value in the range of 0 to 4294967295.

3.1.4. unsigned64

The type "unsigned64" represents a non-negative integer value in the range of 0 to 18446744073709551615.

3.1.5. signed8

The type "signed8" represents an integer value in the range of -128 to 127.

3.1.6. signed16

The type "signed16" represents an integer value in the range of -32768 to 32767.

3.1.7. signed32

The type "signed32" represents an integer value in the range of -2147483648 to 2147483647.

3.1.8. signed64

The type "signed64" represents an integer value in the range of -9223372036854775808 to 9223372036854775807.

3.1.9. float32

The type "float32" corresponds to an IEEE single-precision 32-bit floating point type as defined in [IEEE.754.1985].

3.1.10. float64

The type "float64" corresponds to an IEEE double-precision 64-bit floating point type as defined in [IEEE.754.1985].

3.1.11. boolean

The type "boolean" represents a binary value. The only allowed values are "true" and "false".

3.1.12. macAddress

The type "macAddress" represents a MAC-48 address as in [IEEE.802-3.2002].

3.1.13. octetArray

The type "octetArray" represents a finite-length string of octets.

3.1.14. string

The type "string" represents a finite-length string of valid characters from the Unicode coded character set [ISO.10646]. Unicode incorporates ASCII [RFC20] and the characters of many other international character sets.

3.1.15. dateTimeSeconds

The data type "dateTimeSeconds" represents a time value expressed with second-level precision.

3.1.16. dateTimeMilliseconds

The data type "dateTimeMilliseconds" represents a time value expressed with millisecond-level precision.

3.1.17. dateTimeMicroseconds

The type "dateTimeMicroseconds" represents a time value expressed with microsecond-level precision.

3.1.18. `dateTimeNanoseconds`

The type "`dateTimeNanoseconds`" represents a time value expressed with nanosecond-level precision.

3.1.19. `ipv4Address`

The type "`ipv4Address`" represents an IPv4 address.

3.1.20. `ipv6Address`

The type "`ipv6Address`" represents an IPv6 address.

3.1.21. `basicList`

The type "`basicList`" supports structured data export as described in [RFC6313]; see section 4.5.1 of that document for encoding details.

3.1.22. `subTemplateList`

The type "`subTemplateList`" supports structured data export as described in [RFC6313]; see section 4.5.2 of that document for encoding details.

3.1.23. `subTemplateMultiList`

The type "`subTemplateMultiList`" supports structured data export as described in [RFC6313]; see section 4.5.3 of that document for encoding details.

3.2. Data Type Semantics

This section describes the set of valid data type semantics of the IPFIX information model. A sub-registry of data type semantics [IPFIX-IANA] is established in [RFC5610]; the restrictions on the use of semantics below are compatible with those specified in section 3.10 of that document. These semantics apply only to numeric types, as noted in the description of each semantic below.

Further data type semantics may be specified by future updates to this document. Changes to the associated IPFIX Information Element Semantics sub-registry [IPFIX-IANA] require a Standards Action [RFC5226].

3.2.1. `quantity`

A numeric (integral or floating point) value representing a measured value pertaining to the record. This is distinguished from counters

that represent an ongoing measured value whose "odometer" reading is captured as part of a given record. This is the default semantic type of all numeric data types.

3.2.2. totalCounter

An integral value reporting the value of a counter. Counters are unsigned and wrap back to zero after reaching the limit of the type. For example, an unsigned64 with counter semantics will continue to increment until reaching the value of $2^{64} - 1$. At this point, the next increment will wrap its value to zero and continue counting from zero. The semantics of a total counter is similar to the semantics of counters used in SNMP, such as Counter32 defined in [RFC2578]. The only difference between total counters and counters used in SNMP is that the total counters have an initial value of 0. A total counter counts independently of the export of its value.

3.2.3. deltaCounter

An integral value reporting the value of a counter. Counters are unsigned and wrap back to zero after reaching the limit of the type. For example, an unsigned64 with counter semantics will continue to increment until reaching the value of $2^{64} - 1$. At this point, the next increment will wrap its value to zero and continue counting from zero. The semantics of a delta counter is similar to the semantics of counters used in SNMP, such as Counter32 defined in RFC 2578 [RFC2578]. The only difference between delta counters and counters used in SNMP is that the delta counters have an initial value of 0. A delta counter is reset to 0 each time it is exported and/or expires without export.

3.2.4. identifier

An integral value that serves as an identifier. Specifically, mathematical operations on two identifiers (aside from the equality operation) are meaningless. For example, Autonomous System ID 1 * Autonomous System ID 2 is meaningless. Identifiers MUST be one of the signed or unsigned data types.

3.2.5. flags

An integral value that represents a set of bit fields. Logical operations are appropriate on such values, but not other mathematical operations. Flags MUST always be of an unsigned data type.

4. Information Element Identifiers

All Information Elements defined in the IANA IPFIX Information Element registry [IPFIX-IANA] have their identifiers assigned by IANA.

The value of these identifiers is in the range of 1-32767. Within this range, Information Element identifier values in the sub-range of 1-127 are compatible with field types used by NetFlow version 9 [RFC3954] for historical reasons.

In general, IANA will add newly registered Information Elements to the registry, assigning the lowest available Information Element identifier in the range 128-32767.

Enterprise-specific Information Element identifiers have the same range of 1-32767, but they are coupled with an additional enterprise identifier. For enterprise-specific Information Elements, Information Element identifier 0 is also reserved. Enterprise-specific Information Element identifiers can be chosen by an enterprise arbitrarily within the range of 1-32767. The same identifier may be assigned by other enterprises for different purposes; these Information Elements are distinct because the Information Element identifier is coupled with an enterprise identifier.

Enterprise identifiers are to be registered as SMI network management private enterprise code numbers with IANA. The registry can be found at [PEN-IANA].

5. Information Elements

[IPFIX-IANA] is now the normative reference for IPFIX Information Elements. At the time of publication of [RFC5102], this section defined the initial contents of that registry.

As a historical note, Information Elements were organized into categories in [RFC5102] according to their semantics and their applicability; these categories were not carried forward into [IPFIX-IANA] as an organizing principle. The categories (with example IEs) were:

1. Identifiers (e.g. ingressInterface)
2. Metering and Exporting Process Configuration (e.g. exporterIPv4Address)
3. Metering and Exporting Process Statistics (e.g. exportedOctetTotalCount)
4. IP Header Fields (e.g. sourceIPv4Address)
5. Transport Header Fields (e.g. sourceTransportPort)
6. Sub-IP Header Fields (e.g. sourceMacAddress)
7. Derived Packet Properties (e.g. bgpSourceAsNumber)
8. Min/Max Flow Properties (e.g. minimumIpTotalLength)
9. Flow Timestamps (e.g. flowStartTimeMilliseconds)
10. Per-Flow Counters (e.g. octetDeltaCount)
11. Miscellaneous Flow Properties (e.g. flowEndReason)
12. Padding (paddingOctets)

Information Elements derived from fields of packets or from packet treatment can typically serve as Flow Keys used for mapping packets to Flows. These Information Elements were placed in categories 4-7 in the original categorization.

Information Elements not serving as Flow Keys may have different values for each packet in a Flow. For Information Elements with values derived from packets fields or packet treatment, and for which the value may change from packet to packet within a single Flow, the exported value of an Information Element is by default determined by the first packet observed for the corresponding Flow; the description of the Information Element may however explicitly specify different semantics. This simple rule allows writing all Information Elements related to header fields once when the first packet of the Flow is observed. For further observed packets of the same Flow, only Flow properties that depend on more than one packet need to be updated; these Information Elements were placed in categories 8-11 in the original categorization.

Information Elements with a name having the "post" prefix (e.g. postIpClassOfService), do not necessarily report properties that were actually observed at the Observation Point, but may be retrieved by other means within the Observation Domain. These Information Elements can be used if there are middlebox functions within the Observation Domain changing Flow properties after packets passed the Observation Point; they may also be reported directly by the Observation Point if the Observation Point is situated such as to observe packets on both sides of the middlebox.

6. Extending the Information Model

A key requirement for IPFIX is to allow for extension of the Information Model via the IANA IPFIX registry [IPFIX-IANA]. New Information Element definitions can be added to this registry subject to an Expert Review [RFC5226], with additional process considerations described in [IPFIX-IE-DOCTORS]; that document also provides guidelines for authors and reviewers of new Information Element definitions.

For new Information Elements, the type space defined in Section 3 can be used. If required, new abstract data types can be added to the data type subregistry [IPFIX-IANA] defined in [RFC5610]. New abstract data types and semantics are subject to Standards Action [RFC5226], and MUST be defined in IETF Standards Track documents updating this document.

Enterprises may wish to define Information Elements without registering them with IANA. IPFIX explicitly supports enterprise-specific Information Elements. Enterprise-specific Information Elements are described in Sections 2.1 and 4; guidelines for using them appear in [IPFIX-IE-DOCTORS].

7. IANA Considerations

As this document obsoletes [RFC5102], upon publication of this document, IANA will update the Reference to the IPFIX Information Element registry [IPFIX-IANA], the IPFIX MPLS Label Type subregistry of that registry, the urn:ietf:params:xml:ns:ipfix-info XML namespace, and the urn:ietf:params:xml:schema:ipfix-info XML schema to refer to this document.

However, [RFC5102] still provides a historical reference for the initial entries in the IPFIX Information Element registry. Therefore, IANA will keep [RFC5102] as the Requestor of those Information Elements in the IPFIX Information Element registry which list [RFC5102] as their Requestor, and add the following explanatory note to the IPFIX Information Element registry upon publication of this document:

"RFC XXXX has obsoleted RFC 5102; references to RFC 5102 in this registry remain as part of the historical record."

The Information Element Specification Template in Section 2.1 contains two new columns not present in [RFC5102]. On publication of this document, IANA will create a new Revision column in the IPFIX Information Element Registry, and set the Revision of existing Information Elements to 0. IANA will also create a new Date column in

the IPFIX Information Element Registry, and set the Date of all existing Information Elements to the publication date of this document.

To identify Information Elements with identifiers 127 or below as NetFlow v9 [RFC3954] compatible, upon publication of this document, IANA will set the Name of all existing Reserved Information Elements with identifier 127 or less to "Assigned for NetFlow v9 compatibility", and the Reference of those Information Elements to [RFC3954].

As IANA now has change control of the schema used for the IANA IPFIX Information Element Registry [IPFIX-IANA], IANA will deprecate the previous XML Schema for the description of Information Elements `urn:ietf:params:xml:schema:ipfix-info` [IPFIX-XML-SCHEMA].

To support the process described in Section 7.4, IANA will establish a mailing list for communicating with the IE-DOCTORS experts, named `ie-doctors@ietf.org`.

The remaining subsections of this section contain no actions for IANA.

7.1. IPFIX Information Elements

This document refers to Information Elements, for which the Internet Assigned Numbers Authority (IANA) has created the IPFIX Information Element Registry [IPFIX-IANA]. The columns of this registry must at minimum be able to store the information defined in the template in Section 2.1; it may contain other information as necessary for the management of the registry.

The process for making additions or other changes to the IPFIX Information Element Registry is given in Section 7.4.

7.2. MPLS Label Type Identifier

Information Element #46, named `mplsTopLabelType`, carries MPLS label types. Values for 5 different types have initially been defined. For ensuring extensibility of this information, IANA has created a new subregistry for MPLS label types and filled it with the initial list from the description Information Element #46, `mplsTopLabelType`.

New assignments for MPLS label types are administered by IANA through Expert Review [RFC5226], i.e., review by one of a group of experts designated by an IETF Area Director. The group of experts must double check the label type definitions with already defined label types for completeness, accuracy, and redundancy. The specification

of new MPLS label types MUST be published using a well-established and persistent publication medium.

7.3. XML Namespace and Schema

The prior version of this document [RFC5102] specified an XML schema for IPFIX Information Element definitions [IPFIX-XML-SCHEMA], which was used in the generation of the document text itself. When the IANA IPFIX Information Element registry [IPFIX-IANA] was created, change control on the registry and the schema used to validate it passed to IANA.

The use of a machine-readable syntax for the registry enables the creation of IPFIX tools that can automatically adapt to extensions to the information model. It should be noted that the use of XML in Exporters, Collectors, or other tools is not mandatory for the deployment of IPFIX. In particular, Exporting Processes do not produce or consume XML as part of their operation. IPFIX Collectors MAY take advantage of the machine-readability of the information model vs. hard coding their behavior or inventing proprietary means for accommodating extensions. However, Collectors SHOULD NOT poll the IANA registry [IPFIX-IANA] directly at runtime, in order to avoid unnecessary load on the IANA infrastructure serving the registry.

The reference to the current schema is embedded in the registry [IPFIX-IANA]; this schema may change from time to time as necessary to support the maintenance of the registry. As such, the schema urn:ietf:params:xml:schema:ipfix-info [IPFIX-XML-SCHEMA] specified in [RFC5102] has been deprecated.

7.4. Addition, Revision, and Deprecation

New assignments for IPFIX Information Elements are administered by IANA through Expert Review [RFC5226]. These experts are referred to as IE-DOCTORS experts, and are appointed by the IESG. The process they follow is defined in [IPFIX-IE-DOCTORS].

Information Element identifiers in the range 1-127 are compatible with field types used by NetFlow version 9 [RFC3954] for historical reasons, and must not be assigned unless the Information Element is compatible with the NetFlow version 9 protocol, as determined by an IE-DOCTORS expert designated by the IESG as a Netflow version 9 expert.

Future assignments added to the IPFIX Information Element Registry which require subregistries for enumerated values (e.g. section 7.2, below) must have those subregistries added simultaneously with the new assignment; additions to these subregistries must be subject to

Expert Review [RFC5226]. Unless specified at assignment time, the experts for the subregistry will be the same as for the Information Element registry as a whole.

When IANA receives a request to add, revise, or deprecate an Information Element in the IPFIX Information Elements Registry, it forwards the request to the IE-DOCTORS experts for review.

When IANA receives an approval for a request to add an Information Element definition from the IE-DOCTORS experts, it adds that Information Element to the registry. The approved request may include changes made by the requestor and/or reviewers as compared to the original request.

When IANA receives an approval for a request to revise an Information Element definition from the IE-DOCTORS experts, it changes that Information Element's definition in the registry, and updates the Revision and Date columns as appropriate. The approved request may include changes from the original request. If the original Information Element was added to the registry with IETF consensus (i.e., was defined by an RFC), the revision will require IETF consensus as well.

When IANA receives an approval for a request to deprecate an Information Element definition from the IE-DOCTORS experts, it changes that Information Element's definition in the registry, and updates the Revision and Date columns as appropriate. The approved request may include changes from the original request. If the original Information Element was added to the registry with IETF consensus (i.e., was defined by an RFC), the deprecation will require IETF consensus as well.

8. Security Considerations

The IPFIX information model itself does not directly introduce security issues. Rather, it defines a set of attributes that may for privacy or business issues be considered sensitive information.

For example, exporting values of header fields may make attacks possible for the receiver of this information, which would otherwise only be possible for direct observers of the reported Flows along the data path.

The underlying protocol used to exchange the information described here must therefore apply appropriate procedures to guarantee the integrity and confidentiality of the exported information. These protocols are defined in separate documents, specifically the IPFIX protocol document [RFC5101bis].

9. Acknowledgements

This document is substantially based on [RFC5102]; the editors thank the authors of that document, listed below as contributors. Special thanks to Paul Aitken, for the detailed review.

10. References

10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC6313] Claise, B., Dhandapani, G., Aitken, P, and S. Yates, "Export of Structured Data in IP Flow Information Export (IPFIX)", RFC6313, July 2011.
- [RFC5101bis]
Claise, B., and B. Trammell, Editors, "Specification of the IP Flow Information eXport (IPFIX) Protocol for the Exchange of IP Traffic Flow Information", draft-ietf-ipfix-protocol-rfc5101bis-04, Work in Progress, December 2012.
- [IPFIX-IE-DOCTORS]
Trammell, B., and B. Claise, "Guidelines for Authors and Reviewers of IPFIX Information Elements", draft-ietf-ipfix-ie-doctors-07, Work in Progress, October 2012.

10.2. Informative References

- [IEEE.802-3.2002]
Insitute of Electrical and Electronics Engineers, "Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications", IEEE Standard 802.3, September 2002.
- [IEEE.754.1985]
Institute of Electrical and Electronics Engineers, "Standard for Binary Floating-Point Arithmetic", IEEE Standard 754, August 1985.

- [ISO.10646] International Organization for Standardization, "Information technology - Universal Coded Character Set (UCS)", ISO/IEC 10646:2012(E), June 2012.
- [RFC20] V. Cerf, "ASCII format for Network Interchange", RFC 20, October 1969.
- [RFC2578] McCloghrie, K., Perkins, D., and J. Schoenwaelder, "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.
- [RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.
- [RFC3234] Carpenter, B. and S. Brim, "Middleboxes: Taxonomy and Issues", RFC 3234, February 2002.
- [RFC3444] Pras, A. and J. Schoenwaelder, "On the Difference between Information Models and Data Models", RFC 3444, January 2003.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.
- [RFC3917] Quittek, J., Zseby, T., Claise, B., and S. Zander, "Requirements for IP Flow Information Export (IPFIX)", RFC 3917, October 2004.
- [RFC3954] Claise, B., Ed., "Cisco Systems NetFlow Services Export Version 9", RFC 3954, October 2004.
- [RFC5101] Claise, B., Bryant, S., Leinen, S., Dietz, T., and Trammell, B., "Specification of the IPFIX Protocol for the Exchange of IP Traffic Flow Information", RFC 5101, January 2008.
- [RFC5102] Quittek, J., Bryant, S., Claise, B., Aitken, P., and Meyer, J., "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.

- [RFC5153] Boschi, E., Mark, L., Quittek J., and P. Aitken, "IP Flow Information Export (IPFIX) Implementation Guidelines", RFC5153, April 2008.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.
- [RFC5470] Sadasivan, G., Brownlee, N., Claise, B., and J. Quittek, "Architecture for IP Flow Information Export", RFC5470, March 2009.
- [RFC5471] Schmoll, C., Aitken, P., and B. Claise, "Guidelines for IP Flow Information Export (IPFIX) Testing", RFC5471, March 2009.
- [RFC5472] Zseby, T., Boschi, E., Brownlee, N., and B. Claise, "IP Flow Information Export (IPFIX) Applicability", RFC5472, March 2009.
- [RFC5473] Boschi, E., Mark, L., and B. Claise, "Reducing Redundancy in IP Flow Information Export (IPFIX) and Packet Sampling (PSAMP) Reports", RFC5473, March 2009.
- [RFC5610] Boschi, E., Trammell, B., Mark, L., and T. Zseby, "Exporting Type Information for IP Flow Information Export (IPFIX) Information Elements", July 2009.
- [RFC6183] Kobayashi, A., Claise, B., Muenz, G, and K. Ishibashi, "IP Flow Information Export (IPFIX) Mediation: Framework", RFC6183, April 2011.
- [RFC6615] Dietz, T., Kobayashi, A., Claise, B., and G. Muenz, "Definitions of Managed Objects for IP Flow Information Export", RFC6615, June 2012.
- [RFC6728] Muenz, G., Claise, B., and P. Aitken, "Configuration Data Model for IPFIX and PSAMP", RFC 6728, October 2012.
- [IPFIX-MED-PROTO] Claise, B., Kobayashi, A., and B. Trammell, "Operation of the IP Flow Information Export (IPFIX) Protocol on IPFIX Mediators", draft-ietf-ipfix-mediation-protocol-02, Work in Progress, July 2012.

[IPFIX-IANA]

<http://www.iana.org/assignments/ipfix/ipfix.xml>

[PEN-IANA]

<http://www.iana.org/assignments/enterprise-numbers>

[IPFIX-XML-SCHEMA]

<http://www.iana.org/assignments/xml-registry/schema/ipfix.xsd>

Authors' Addresses

Benoit Claise (Ed.)
Cisco Systems
De Kleetlaan 6a b1
1831 Diegem
Belgium

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

Brian Trammell (Ed.)
Swiss Federal Institute of Technology Zurich
Gloriastrasse 35
8092 Zurich
Switzerland

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

Contributors' Addresses

Juergen Quittek
NEC
Kurfuersten-Anlage 36
Heidelberg 69115
Germany

Phone: +49 6221 90511-15
EMail: quittek@nw.neclab.eu
URI: <http://www.neclab.eu/>

Stewart Bryant
Cisco Systems, Inc.
250, Longwater Ave., Green Park
Reading RG2 6GB
United Kingdom

EMail: stbryant@cisco.com

Paul Aitken
Cisco Systems, Inc.
96 Commercial Quay
Edinburgh EH6 6LX
Scotland

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

Jeff Meyer
PayPal
2211 N. First St.
San Jose, CA 95131-2021
US

Phone: +1 408 976-9149
EMail: jemeyer@paypal.com
URI: <http://www.paypal.com>

IPFIX Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 29, 2013

B. Claise
Cisco Systems, Inc.
A. Kobayashi
NTT
B. Trammell
ETH Zurich
February 25, 2013

Operation of the IP Flow Information Export (IPFIX) Protocol on IPFIX
Mediators
draft-ietf-ipfix-mediation-protocol-04.txt

Abstract

This document specifies the operation of the IP Flow Information Export (IPFIX) protocol specific to IPFIX Mediators, including Template and Observation Point management, timing considerations, and other Mediator-specific concerns.

Status of this Memo

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

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

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

This Internet-Draft will expire on August 29, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

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

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. IPFIX Documents Overview	3
1.2. IPFIX Mediator Documents Overview	4
1.3. Relationship with the IPFIX and PSAMP Protocols	5
2. Terminology	5
3. Handling IPFIX Message Headers	8
4. Template Management	10
4.1. Passing Unmodified Templates through an IPFIX Mediator	11
4.1.1. Template Mapping and Information Element Ordering	14
4.2. Creating New Templates at an IPFIX Mediator	15
4.3. Handling Unknown Information Elements	16
5. Preserving Original Observation Point Information	16
5.1. originalExporterIPv4Address Information Element	18
5.2. originalExporterIPv6Address Information Element	18
6. Managing Observation Domain IDs	19
6.1. originalObservationDomainId Information Element	19
7. Timing Considerations	20
8. Transport Considerations	21
9. Collecting Process Considerations	21
10. Specific Reporting Requirements	22
10.1. Intermediate Process Reliability Statistics Template	22
10.2. Flow Key Options Template	23
10.3. intermediateProcessId Information Element	24
10.4. ignoredRecordTotalCount Information Element	24
11. Configuration Management	24
12. Security Considerations	25
13. IANA Considerations	25
14. Acknowledgments	25
15. References	26
15.1. Normative References	26
15.2. Informative References	27
Authors' Addresses	28

1. Introduction

The IPFIX architectural components in [RFC5470] consist of IPFIX Devices and IPFIX Collectors communicating using the IPFIX protocol [I-D.ietf-ipfix-protocol-rfc5101bis], which specifies how to export IP Flow information. This protocol is designed to export information about IP traffic Flows and related measurement data, where a Flow is defined by a set of key attributes (e.g. source and destination IP address, source and destination port, etc.).

However, thanks to its Template mechanism, the IPFIX protocol can export any type of information, as long as the relevant Information Element is specified in the IPFIX Information Model [I-D.ietf-ipfix-information-model-rfc5102bis], registered with IANA, or specified as an enterprise-specific Information Element. The specifications in the IPFIX protocol [I-D.ietf-ipfix-protocol-rfc5101bis] have not been defined in the context of an IPFIX Mediator receiving, aggregating, correlating, anonymizing, etc... Flow Records from the one or multiple Exporters. Indeed, the IPFIX protocol must be adapted for Intermediate Processes, as defined in the IPFIX Mediation Reference Model as specified in Figure A of [RFC6183], which is based on the IPFIX Mediation Problem Statement [RFC5982].

This document specifies the IP Flow Information Export (IPFIX) protocol in the context of the implementation and deployment of IPFIX Mediators. The use of the IPFIX protocol within an IPFIX Mediator -- a device which contains both a Collecting Process and an Exporting Process -- has an impact on the technical details of the usage of the protocol. An overview of the technical problem is covered in section 6 of [RFC5982]: loss of original Exporter information, loss of base time information, transport sessions management, loss of Options Template Information, Template Id management, considerations for network considerations for aggregation.

The specifications in this document are based on the IPFIX protocol specifications [I-D.ietf-ipfix-protocol-rfc5101bis] but adapted according to the IPFIX Mediation Framework [RFC6183].

1.1. IPFIX Documents Overview

The IPFIX Protocol [I-D.ietf-ipfix-protocol-rfc5101bis] provides network administrators with access to IP Flow information.

The architecture for the export of measured IP Flow information out of an IPFIX Exporting Process to a Collecting Process is defined in the IPFIX Architecture [RFC5470], per the requirements defined in the IPFIX Requirement doc, [RFC3917].

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

IPFIX has a formal description of IPFIX Information Elements, their name, type and additional semantic information, as specified in the IPFIX Information Model [I-D.ietf-ipfix-information-model-rfc5102bis]. The registry is maintained by IANA [IPFIX-IANA]. New Information Element definitions can be added to this registry subject to an Expert Review [RFC5226], with additional process considerations described in [IPFIX-IE-DOCTORS]; that document also provides guidelines for authors and reviewers of new Information Element definitions. The inline export of the Information Element type information is specified in [RFC5610].

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.

1.2. IPFIX Mediator Documents Overview

The "IPFIX Mediation: Problem Statement" [RFC5982] provides an overview of the applicability of IPFIX Mediators, and defines requirements for IPFIX 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 IPFIX Mediators within an IPFIX collection infrastructure.

The "IPFIX Mediation: Framework" [RFC6183], which details the IPFIX Mediation reference model and the components of an IPFIX Mediator, provides more architectural details of the arrangement of Intermediate Processes within an IPFIX Mediator.

Documents specifying the operations of specific Intermediate Processes cover the operation of these Processes within the IPFIX Mediator framework, and comply with the specifications given in this document; they may additionally specify the operation of the process independently, outside the context of an IPFIX Mediator, when this is appropriate. 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. As of today, these documents are:

1. "IP Flow Anonymization Support", [RFC6235], which describes Anonymization techniques for IP flow data and the export of

Anonymized data using the IPFIX protocol.

2. "Flow Selection Techniques" [I-D.ietf-ipfix-flow-selection-tech], which describes the process of selecting a subset of Flows from all Flows observed at an Observation Point, the flow selection motivations, and some specific flow selection techniques.
3. "Exporting Aggregated Flow Data using IP Flow Information Export" [I-D.ietf-ipfix-a9n] which describes Aggregated Flow export within the framework of IPFIX Mediators and defines an interoperable, implementation-independent method for Aggregated Flow export.

This document specifies the IP Flow Information Export (IPFIX) protocol specific to Mediation, i.e. the specifications that all Intermediate Processes type must comply to. Some extra specifications might be required per Intermediate Process type (In which case, the Intermediate Process specific document would cover those).

1.3. Relationship with the IPFIX and PSAMP Protocols

The specification in this document applies to the IPFIX protocol specifications [I-D.ietf-ipfix-protocol-rfc5101bis]. All specifications from [I-D.ietf-ipfix-protocol-rfc5101bis] apply unless specified otherwise in this document.

As the Packet Sampling (PSAMP) protocol specifications [RFC5476] are based on the IPFIX protocol specifications, the specifications in this document are also valid for the PSAMP protocol. Therefore, the method specified by this document also applies to PSAMP.

2. Terminology

IPFIX-specific terms, such as Observation Domain, Flow, Flow Key, Metering Process, Exporting Process, Exporter, IPFIX Device, Collecting Process, Collector, Template, IPFIX Message, Message Header, Template Record, Data Record, Options Template Record, Set, Data Set, Information Element, Scope and Transport Session, used in this document are defined in [I-D.ietf-ipfix-protocol-rfc5101bis]. The PSAMP-specific terms used in this document, such as Filtering and Sampling, are defined in [RFC5476].

IPFIX Mediation terms related to aggregation, such as the Interval, Aggregated Flow, and Aggregated Function are defined in [I-D.ietf-ipfix-a9n].

The IPFIX Mediation-specific terminology used in this document is defined in "IPFIX Mediation: Problem Statement" [RFC5982], and reused in "IPFIX Mediation: Framework" [RFC6183]. However, since both of those documents are informational RFCs, the definitions have been reproduced here along with additional definitions.

Similarly, since [RFC6235] is an experimental RFC, the Anonymization Record, Anonymized Data Record, and Intermediate Anonymization Process terms, specified in [RFC6235], are also reproduced here.

In this document, as in [I-D.ietf-ipfix-protocol-rfc5101bis], [RFC5476], [I-D.ietf-ipfix-a9n], and [RFC6235], the first letter of each IPFIX-specific and PSAMP-specific term is capitalized along with the IPFIX Mediation-specific term defined here.

In this document, we call a stream of records carrying flow- or packet-based information a "record stream". The records may be encoded as IPFIX Data Records of any other format.

Transport Session Information: The Transport Session is specified in [I-D.ietf-ipfix-protocol-rfc5101bis]. In SCTP, the Transport Session Information is the SCTP association. In TCP and UDP, the Transport Session Information corresponds to a 5-tuple {Exporter IP address, Collector IP address, Exporter transport port, Collector transport port, transport protocol}.

Original Exporter: An Original Exporter is an IPFIX Device that hosts the Observation Points where the metered IP packets are observed.

Original Observation Point: An Observation Point of the Original Exporter. In the case of the Intermediate Aggregation Process on an IPFIX Mediator, the Original Observation Point can be composed of, but not limited to, a (set of) specific Exporter(s), a (set of) specific interface(s) on an Exporter, a (set of) line card(s) on an Exporter, or any combinations of these.

IPFIX Mediation: IPFIX Mediation is the manipulation and conversion of a record stream for subsequent export using the IPFIX protocol.

Template Mapping: A mapping from Template Records and/or Options Template Records received by an IPFIX Mediator to Template Records and/or Options Template Records sent by that IPFIX Mediator. Each entry in a Template Mapping is scoped by incoming or outgoing Transport Session and Observation Domain, as with Templates and Options Templates in the IPFIX Protocol.

Anonymization Record: A record that defines the properties of the anonymization applied to a single Information Element within a single Template or Options Template, as in [RFC6235].

Anonymized Data Record: A Data Record within a Data Set containing at least one Information Element with Anonymized values. The Information Element(s) within the Template or Options Template describing this Data Record SHOULD have a corresponding Anonymization Record, as in [RFC6235].

The following terms are used in this document to describe the architectural entities used by IPFIX Mediation.

Intermediate Process: An Intermediate Process takes a record stream as its input from Collecting Processes, Metering Processes, IPFIX File Readers, other Intermediate Processes, or other record sources; performs some transformations on this stream, based upon the content of each record, states maintained across multiple records, or other data sources; and passes the transformed record stream as its output to Exporting Processes, IPFIX File Writers, or other Intermediate Processes, in order to perform IPFIX Mediation. Typically, an Intermediate Process is hosted by an IPFIX Mediator. Alternatively, an Intermediate Process may be hosted by an Original Exporter.

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

Specific Intermediate Processes are described below.

Intermediate Conversion Process (as in [RFC6183]): 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 (as in [I-D.ietf-ipfix-a9n]): an Intermediate Process (IAP) as in [RFC6183] that aggregates records, based upon a set of Flow Keys or functions applied to fields from the record.

Intermediate Correlation Process (as in [RFC6183]): 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 Anonymization Process (as in [RFC6235]): An intermediate process that takes Data Records and transforms them into Anonymized Data Records.

Intermediate Selection Process (as in [RFC6183]): 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 Flow Selection Process (as in [I-D.ietf-ipfix-flow-selection-tech]): An Intermediate Flow Selection Process is an Intermediate Process as in [RFC6183] that takes Flow Records as its input and selects a subset of this set as its output. Intermediate Flow Selection Process is a more general concept than Intermediate Selection Process as defined in [RFC6183]. While an Intermediate Selection Process selects Flow Records from a sequence based upon criteria-evaluated Flow record values and passes only those Flow Records that match the criteria, an Intermediate Flow Selection Process selects Flow Records using selection criteria applicable to a larger set of Flow characteristics and information.

3. Handling IPFIX Message Headers

The format of the IPFIX Message Header as exported by an IPFIX Mediator is shown in Figure 1. Note that the format is compatible with the IPFIX Message Header defined in [I-D.ietf-ipfix-protocol-rfc5101bis], with some field definitions (for the example, the Export Time) updated in the context of the IPFIX Mediator.

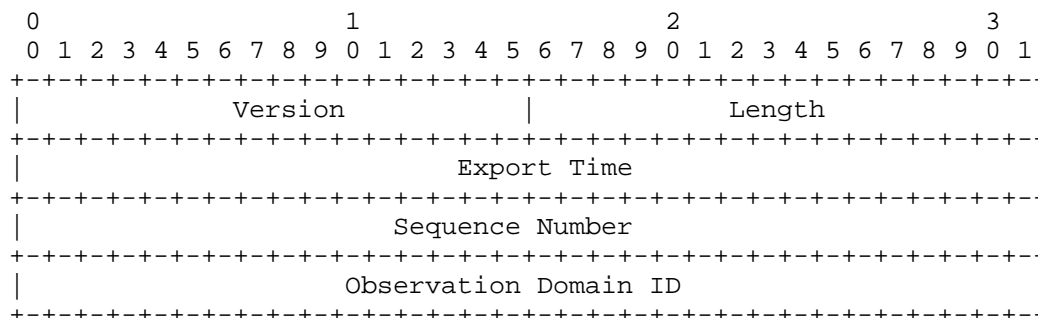


Figure 1: IP Message Header format

The header fields as exported by an IPFIX Mediator are describe below.

Version: Version of IPFIX to which this Message conforms. The value of this field is 0x000a for the current version, incrementing by one the version used in the NetFlow services export version 9 [RFC3954].

Length: Total length of the IPFIX Message, measured in octets, including Message Header and Set(s).

Export Time: Time at which the IPFIX Message Header leaves the IPFIX Mediator, expressed in seconds since the UNIX epoch of 1 January 1970 at 00:00 UTC, encoded as an unsigned 32-bit integer. However, in the specific case of an IPFIX Mediator containing an Intermediate Conversion Process, the IPFIX Mediator MAY keep the export time received from the incoming Transport Session.

Sequence Number: Incremental sequence counter modulo 2^{32} of all IPFIX Data Records sent in a the current stream from the current Observation Domain by the Exporting Process. Each SCTP Stream counts sequence numbers separately, while all messages in a TCP connection or UDP transport session are considered to be part of the same stream. This value SHOULD be used by the Collecting Process to identify whether any IPFIX Data Records have been missed. Template and Options Template Records do not increase the Sequence Number.

Observation Domain ID: A 32-bit identifier of the Observation Domain that is locally unique to the Exporting Process. The Exporting Process uses the Observation Domain ID to uniquely identify to the Collecting Process the Observation Domain that metered the Flows. It is RECOMMENDED that this identifier also be unique per IPFIX Device. Collecting Processes SHOULD use the

Transport Session and the Observation Domain ID field to separate different export streams originating from the same Exporter. The Observation Domain ID SHOULD be 0 when no specific Observation Domain ID is relevant for the entire IPFIX Message, for example, when exporting the Exporting Process Statistics, or in case of a hierarchy of Collectors when aggregated Data Records are exported. See Section 4.1 for special considerations for Observation Domain management while passing unmodified templates through an IPFIX Mediator, and Section 5 for guidelines for preservation of original Observation Domain information at an IPFIX Mediator.

The following specifications, copied over from [I-D.ietf-ipfix-protocol-rfc5101bis] have some implications in this document: "Template Withdrawals MAY appear interleaved with Template Sets, Options Template Sets, and Data Sets within an IPFIX Message. In this case, the Templates and Template Withdrawals shall be taken to take effect in the order in which they appear in the IPFIX Message."

If an IPFIX Mediator receives an IPFIX Message composed of Template Withdrawals and Template Sets, and if the IPFIX Mediator forwards this IPFIX Message, it MUST not modify the Set order. Note that the Template Mapping (see section 4.1) is the authoritative source of information on the IPFIX Mediator to decide whether the entire IPFIX Messages can be forwarded as such.

4. Template Management

How an IPFIX Mediator handles the Templates it receives from the Original Exporter depends entirely on the nature of the Intermediate Process running on that IPFIX Mediator.

IPFIX Mediators that pass substantially the same Data Records from the Original Exporter downstream, (e.g., an Intermediate Selection Process), pass unmodified Template as described in Section 4.1; this section describes a Template Mapping required to make this work in the general case, and the correlation between the received and generated IPFIX Message Withdrawals.

IPFIX Mediators that export Data Records which are substantially changed from the Data Records received from the Original Exporter follow the guidelines in Section 4.2 instead: in this case, the IPFIX Mediator generates new (Options) Template Records as a result of the Intermediate Process, and no Template Mapping is required.

Subsequent subsections deal with specific issues in Template management that may occur at IPFIX Mediators.

4.1. Passing Unmodified Templates through an IPFIX Mediator

The first case is a situation where the IPFIX Mediator doesn't modify the (Options) Template Record(s) content. A typical example is an Intermediate Flow Selection Process acting as distributor, which collects Flow Records from one or more Exporters, and based on the Information Elements content, redirects the Flow Records to the appropriate Collector. This example is a typical case of a single network operation center managing multiple universities: an unique IPFIX Collector collects all Flow Records for the common infrastructure, but might be re-exporting specific university Flow Records to the responsible system administrator.

As specified in [I-D.ietf-ipfix-protocol-rfc5101bis], the Template IDs are unique per Exporter, per Transport Session, and per Observation Domain. As there is no guarantee that, for similar Template Records, the Template IDs received on the incoming Transport Session and exported to the outgoing Transport Session would be same, the IPFIX Mediator MUST maintain a Template Mapping composed of related received and exported (Options) Template Records:

- o for each received (Options) Template Record: Template Record Information Elements, Template ID, Observation Domain Id, and Transport Session Information, metadata scoped to the Template (*)
- o for each exported (Options) Template Record: Template Record Information Elements, Template ID, Collector, Observation Domain Id, and Transport Session Information metadata scoped to the Template (*)

(*) The "metadata scoped to the Template" encompasses the metadata, that are scoped to the Template, and that help to determine the semantics of the Template Record. Note that these metadata are typically sent in Data Records described by an Options Template. A example is the flowKeyIndicator: An IPFIX Mediator could potentially received two different Template IDs, from the same Exporter, with the same Information Elements, but with a different set of Flow Keys (indicated by the flowKeyIndicator in an Options Template Record). Another example is the combination of anonymizationFlags and anonymizationTechnique [RFC6235]). This metadata information must be present in the Template Mapping, to stress that the two Template Record semantics are different.

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

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

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

For example, Figure 2 displays an example of an Intermediate Flow Selection Process, re-distributing Data Records to Collectors on the basis of customer networks, i.e. the Route Distinguisher (RD). In this example, the Template Record received from the Exporter #1 is reused towards Collector #1, Collector #2, and Collector #3.

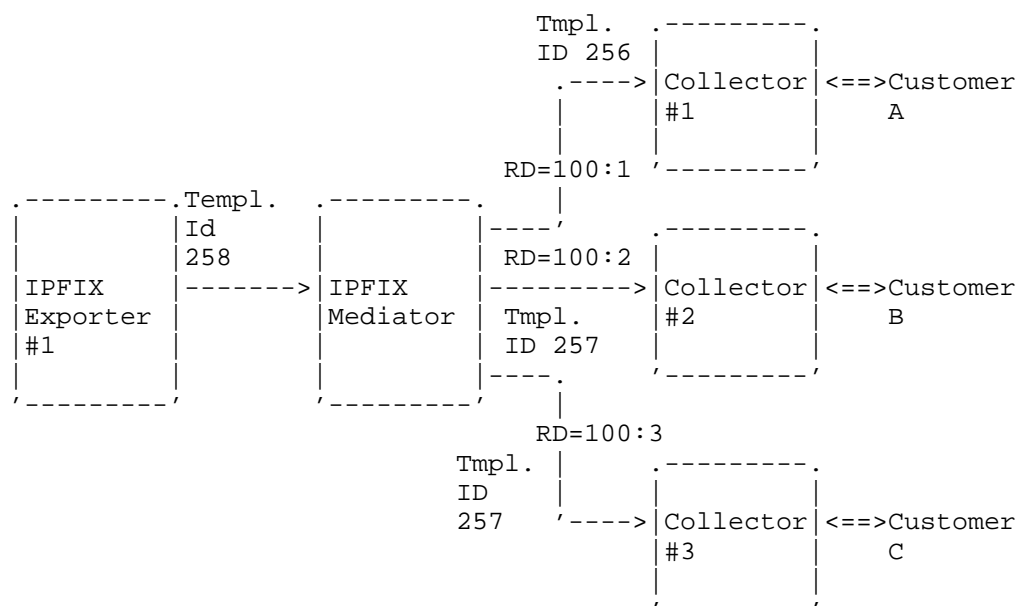


Figure 2: Intermediate Flow Selection Process example

Figure 3 shows the Template Mapping for the system shown in Figure 2.

Template Entry A:
Incoming Transport Session Information (from Exporter#1):
 Source IP: <Exporter#1 export IP address>
 Destination IP: <IPFIX Mediator IP address>
 Protocol: SCTP
 Source Port: <source port>
 Destination Port: 4739 (IPFIX)
Observation Domain Id: <Observation Domain ID>
Template Id: 258
Metadata scoped to the Template : <not applicable in this case>

Template Entry B:
Outgoing Transport Session Information (to Collector#1):
 Source IP: <IPFIX Mediator IP address>
 Destination IP: <IPFIX Collector#1 IP address>
 Protocol: SCTP
 Source Port: <source port>
 Destination Port: 4739 (IPFIX)
Observation Domain Id: <Observation Domain ID>
Template Id: 256
Metadata scoped to the Template : <not applicable in this case>

Template Entry C:
Outgoing Transport Session Information (to Collector#2):
 Source IP: <IPFIX Mediator IP address>
 Destination IP: <IPFIX Collector#2 IP address>
 Protocol: SCTP
 Source Port: <source port>
 Destination Port: 4739 (IPFIX)
Observation Domain Id: <Observation Domain ID>
Template Id: 257
Metadata scoped to the Template : <not applicable in this case>

Template Entry D:
Outgoing Transport Session Information (to Collector#3):
 Source IP: <IPFIX Mediator IP address>
 Destination IP: <IPFIX Collector#3 IP address>
 Protocol: SCTP
 Source Port: <source port>
 Destination Port: 4739 (IPFIX)
Observation Domain Id: <Observation Domain ID>
Template Id: 257
Metadata scoped to the Template : <not applicable in this case>

Figure 3: Template Mapping example: templates

The Template Mapping corresponding to figure B can be displayed as:

```

Template Entry A  <----> Template Entry B
Template Entry A  <----> Template Entry C
Template Entry A  <----> Template Entry D

```

Template Mapping example: mappings

Alternatively, the Template Mapping may be optimized as:

```

      +--> Template Entry B
      |
Template Entry A  <--+--> Template Entry C
      |
      +--> Template Entry D

```

Template Mapping example: mappings

Note that all examples use Transport Sessions based on the SCTP protocol, as simplified use cases. However, the protocol would be important in situations such as an Intermediate Conversion Process doing transport protocol conversion.

4.1.1. Template Mapping and Information Element Ordering

In the situation where Original Exporters each export an (Options) Template to a single IPFIX Mediator, and the (Options) Template Record contains the same Information Elements but in different order, should the IPFIX Mediator maintain a Template Mapping with a single Export Template Record (see figure "Template Mapping and Ordering: a single Export Template Record") or should the IPFIX Mediator maintain multiple independent Template Records (see figure "Template Mapping and Ordering: multiple Export Template Record") before re-exporting to the Collector?

```

Template Entry A  <--+
      |
Template Entry B  <--+--> Template Entry D
      |
Template Entry C  <--+

```

Template Mapping and Ordering: a single Export Template Record

```

Template Entry A  <--+--> Template Entry D
Template Entry B  <--+--> Template Entry E
Template Entry C  <--+--> Template Entry F

```

Template Mapping and Ordering: multiple Export Template Records

The answer depends whether the order of the Information Elements implies some specific semantic. One of the guiding principles in IPFIX protocol specifications that the semantic meaning of one Information Element doesn't depend on the value of the different Information Element. However, there is one noticeable exception, as mentioned in [I-D.ietf-ipfix-protocol-rfc5101bis]:

"Multiple Scope Fields MAY be present in the Options Template Record, in which case, the composite scope is the combination of the scopes. For example, if the two scopes are meteringProcessId and templateId, the combined scope is this Template for this Metering Process. If a different order of Scope Fields would result in a Record having a different semantic meaning, then the order of Scope Fields MUST be preserved by the Exporting Process. For example, in the context of PSAMP [RFC5476], if the first scope defines the filtering function, while the second scope defines the sampling function, the order of the scope is important. Applying the sampling function first, followed by the filtering function, would lead to potentially different Data Records than applying the filtering function first, followed by the sampling function."

If an IPFIX Mediator receives, from multiple Exporters, Template Records with identical Information Elements, but ordered differently, it SHOULD consider those Template Records as identical.

If an IPFIX Mediator receives, from multiple Exporters, Options Template Records with identical and ordered Information Elements in the Scope fields, and with identical Information Elements, but ordered differently, in the non Scope fields, it SHOULD consider those Template Records as identical.

If an IPFIX Mediator receives, from multiple Exporters, Options Template Records with identical Information Elements in the scope, but ordered differently, it MUST consider those Template Records as semantically different.

4.2. Creating New Templates at an IPFIX Mediator

The second case is a situation where the IPFIX Mediator generates new (Options) Template Records as a result of the Intermediate Process.

In this situation, the IPFIX Mediator doesn't need to maintain a Template Mapping, as it generates its own series of (Options) Template Records. However, the following special case might still require a Template Mapping, i.e. a situation where the IPFIX Mediator, typically containing an Intermediate Conversion Process,

Intermediate Aggregation Process, or Intermediate Anonymization Process in case of black-marker Anonymization [RFC6235], generates new (Options) Template Records based on what it receives from the Exporter(s), and based on the Intermediate Process function. In such a case, it's important to keep the correlation between the received (Options) Template Records and exported Derived (Options) Template Records in the Template Mapping. These Template Mappings would be kept as in Section 4.1, except that the exported Template would not be identical to the received Template.

4.3. Handling Unknown Information Elements

Depending on application requirements, Mediators which do not generate new Records SHOULD re-export values for unknown Information Elements, whether enterprise-specific Information Elements or Information Elements in the IANA IPFIX Information Element registry added since the Mediator was implemented or updated. However, as there may be presence or ordering dependencies among the unknown Information Elements, the Mediator MUST NOT omit fields from such re-exported Records, or re-order any fields within the Records.

Mediators which generate new Records, as in Section 4.2, SHOULD NOT use values of Information Elements they do not understand. If they do pass such values, they MUST NOT pass values of unknown Information Elements unless all such values are passed on in the original order in which they were received.

In any case, Mediators handling unknown Information Elements SHOULD log this fact, as it is likely that mediation of records containing unknown values will have unintended consequences.

5. Preserving Original Observation Point Information

Depending on the use case, the Collector in an Exporter - IPFIX Mediator - Collector structure may need to receive information about the Original Observation Point(s), otherwise it may wrongly conclude that the IPFIX Device exporting the Flow Records, i.e. the IPFIX Mediator, directly observed the packets that generated the Flow Records. Two new Information Elements are introduced in the subsections below to address this use case: `originalExporterIPv4Address` and `originalExporterIPv6Address`. Practically, the Original Exporters will not export these Information Elements. Therefore, the Intermediate Process SHOULD report the Original Observation Point(s) to the best of its knowledge. Note that the Configuration Data Model for IPFIX and PSAMP [RFC6728] may help.

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

- o A single Original Exporter (represented by the originalExporterIPv4Address or originalExporterIPv6Address Information Elements)
- o A list of Original Exporters (represented by the originalExporterIPv4Address or originalExporterIPv6Address Information Elements).
- o Any combination or list of Information Elements representing Observation Points. For example:
 - * A list of Original Exporter interface(s) (represented by the originalExporterIPv4Address or originalExporterIPv6Address, the ingressInterface and/or egressInterface Information Elements, respectively)
 - * A list of Original Exporter line card (represented by the originalExporterIPv4Address or originalExporterIPv6Address, the lineCardId Information Elements, respectively)

Some Information Elements characterizing the Observation Point may be added. For example, the flowDirection Information Element specifies the direction of the observation, and, as such, characterizes the Observation Point.

Any combination of the above representations is possible. For example, in case of an Intermediate Aggregation Process, an Original Observation Point could be composed of:

```
exporterIPv4Address 192.0.2.1
exporterIPv4Address 192.0.2.2,
  interface ethernet 0, direction ingress
  interface ethernet 1, direction ingress
  interface serial 1, direction egress
  interface serial 2, direction egress
exporterIPv4Address 192.0.2.3,
  lineCardId 1, direction ingress
```

Figure 4: Complex Observation Point Definition Example

If the Original Observation Point is composed of a list, then the IPFIX Structured Data [RFC6313] MUST be used to export it from the IPFIX Mediator.

The most generic way to export the Original Observation Point is to

use a `subTemplateMultiList`, with the semantic "exactlyOneOf". Taking the previous example, the following encoding can be used:

```
Template Record 257: exporterIPv4Address
Template Record 258: exporterIPv4Address,
                    basicList of ingressInterface, flowDirection
Template Record 259: exporterIPv4Address, lineCardId, flowDirection
```

Figure 5: Complex Observation Point Definition Example: Templates

The Original Observation Point is modeled with the Data Records corresponding to either Template Record 1, Template Record 2, or Template Record 3 but not more than one of these ("exactlyOneOf" semantic). This implies that the Flow was observed at exactly one of the Observation Points reported.

When an IPFIX Mediator receives Flow Records containing the Original Observation Point Information Element, i.e. `originalExporterIPv6Address` or `originalExporterIPv4Address`, the IPFIX Mediator SHOULD NOT modify its value(s) when composing new Flow Records in the general case. Known exceptions include anonymization per [RFC6235] section 7.2.4 and an Intermediate Correlation Process rewriting addresses across NAT. In other words, the Original Observation Point should not be replaced with the IPFIX Mediator Observation Point. The daisy chain of (Exporter, Observation Point) representing the path the Flow Records took from the Exporter to the top Collector in the Exporter - IPFIX Mediator(s) - Collector structure model is out of the scope of this specification.

5.1. `originalExporterIPv4Address` Information Element

Description: The IPv4 address used by the Exporting Process on an Original Exporter, as seen by the Collecting Process on an IPFIX Mediator. Used to provide information about the Original Observation Points to a downstream Collector.

Data Type: `ipv4Address`

ElementId: TBD1

5.2. `originalExporterIPv6Address` Information Element

Description: The IPv6 address used by the Exporting Process on an Original Exporter, as seen by the Collecting Process on an IPFIX Mediator. Used to provide information about the Original Observation Points to a downstream Collector.

Data Type: ipv6Address

ElementId: TBD2

6. Managing Observation Domain IDs

In any case, the Observation Domain ID of any IPFIX Message containing Flow Records relevant to no particular Observation Domain, or to multiple Observation Domains, MUST have an Observation Domain ID of 0, as in Section 3 above, and section 3.1 of [I-D.ietf-ipfix-protocol-rfc5101bis].

IPFIX Mediators that do not change (Options) Template Records MUST maintain a Template Mapping, as detailed in Section 4.1, to ensure that the combination of Observation Domain IDs and Template IDs do not collide on export.

For IPFIX Mediators that export New (Options) Template Records, as in Section 4.2, there are two options for Observation Domain ID management. The first and simplest of these is to completely decouple exported Observation Domain IDs from received Observation Domain IDs; the IPFIX Mediator, in this case, comprises its own set of Observation Domain(s) independent of the Observation Domain(s) of the Original Exporters.

The second option is to provide or maintain a Template Mapping for received (Options) Template Records and exported inferred (Options) Template Records, along with the appropriate Observation Domain IDs per Transport Session, which ensures that the combination of Observation Domain IDs and Template IDs do not collide on export.

In some cases where the IPFIX Message Header can't contain a consistent Observation Domain for the entire IPFIX Message, but the Flow Records exported from the IPFIX Mediator should anyway contain the Observation Domain of the Original Exporter, the (Options) Template Record must contain the originalObservationDomainId Information Element. When an IPFIX Mediator receives Flow Records containing the originalObservationDomainId Information Element, the IPFIX Mediator MUST NOT modify its value(s) when composing new Flow Records with the originalObservationDomainId Information Element.

6.1. originalObservationDomainId Information Element

Description: The Observation Domain ID reported by the Exporting Process on an Original Exporter, as seen by the Collecting Process on an IPFIX Mediator. Used to provide information about the Original Observation Domain to a downstream Collector.

Data Type: unsigned32

Data Type Semantics: identifier

ElementId: TBD3

7. Timing Considerations

The IPFIX Message Header "Export Time" field is the time in seconds since 0000 UTC Jan 1, 1970, at which the IPFIX Message leaves the IPFIX Mediator. However, in the specific case of an IPFIX Mediator containing an Intermediate Conversion Process, the IPFIX Mediator MAY keep the export time received from the incoming Transport Session.

It is RECOMMENDED that IPFIX Mediators handle time using absolute timestamps (e.g. flowStartSeconds, flowStartMilliseconds, flowStartNanoseconds), which are specified relative to the UNIX epoch (00:00 UTC 1 Jan 1970), where possible, rather than relative timestamps (e.g. flowStartSysUpTime, flowStartDeltaMicroseconds), which are specified relative to protocol structures such as system initialization or message export time.

The latter are difficult to manage for two reasons. First, they require constant translation, as the system initialization time of an intermediate system and the export time of an intermediate message will change across mediation operations. Further, relative timestamps introduce range problems. For example, when using the flowStartDeltaMicroseconds and flowEndDeltaMicroseconds Information Elements [iana-ipfix-assignments], the Data Record must be exported within a maximum of 71 minutes after its creation. Otherwise, the 32-bit counter would not be sufficient to contain the flow start time offset. Those time constraints might be incompatible with some of the application requirements of some Intermediate Processes.

Intermediate Processes MUST NOT assume that received records appear in flowStartTime, flowEndTime, or observationTime order. An Intermediate Process processing timing information (e.g., an Intermediate Aggregation Process) MAY ignore records that are significantly out of order, in order to meet application-specific state and latency requirements, but SHOULD report that records were dropped.

When an Intermediate Process aggregates information from different Flow Records, the timestamps on exported records SHOULD be the minimum of the start times and the maximum of the end times in the general case. However, if the Flow Records do not overlap, i.e. if there is a time gap between the times in the Flow Records, then the report may be inaccurate. The IPFIX Mediator is only reporting what it knows, on the basis of the information made available to it - and there may not have been any data to observe during the gap. Then again, if there is an overlap in timestamps, there's the potential of double-accounting: different Observation Points may have observed the same traffic simultaneously. Therefore, as there is not a single rule that fits all different situations, a complete specification of the precise rules of applying Flow Record timestamps at IPFIX Mediators is out of the scope of this document.

Note that [I-D.ietf-ipfix-a9n] provides additional specifications for handling of timestamps at an Intermediate Aggregation Process.

8. Transport Considerations

SCTP [RFC4960] using the PR-SCTP extension specified in [RFC3758] MUST be implemented by all compliant IPFIX Mediator implementations. TCP [RFC0793] MAY also be implemented by IPFIX Mediator compliant implementations. UDP [RFC0768] MAY also be implemented by compliant IPFIX Mediator implementations. Transport-specific considerations for IPFIX Exporters as specified in sections 8.3, 8.4, 9.1, 9.2, and 10 of [I-D.ietf-ipfix-protocol-rfc5101bis] apply to IPFIX Mediators as well.

SCTP SHOULD be used in deployments where IPFIX Mediators and Collectors are communicating over links that are susceptible to congestion. SCTP is capable of providing any required degree of reliability. TCP MAY be used in deployments where IPFIX Mediators and Collectors communicate over links that are susceptible to congestion, but SCTP is preferred due to its ability to limit back pressure on Exporters and its message versus stream orientation. UDP MAY be used, although it is not a congestion-aware protocol. However, in this case, the IPFIX traffic between IPFIX Mediator and Collector MUST run in an environment where IPFIX traffic has been provisioned for, or is contained through some other means.

9. Collecting Process Considerations

Any Collecting Process compliant with [I-D.ietf-ipfix-protocol-rfc5101bis] can receive IPFIX Messages from an IPFIX Mediator. If the IPFIX Mediator uses IPFIX Structured Data

[RFC6313] to export Original Exporter Information as in Section 5, the Collecting Process MUST support [RFC6313].

10. Specific Reporting Requirements

IPFIX provides Options Templates for the reporting on the reliability of processes within the IPFIX Architecture. As each Mediator includes at least one IPFIX Exporting Process, they SHOULD use the Exporting Process Reliability Statistics Options Template, as specified in [I-D.ietf-ipfix-protocol-rfc5101bis].

Analogous to the Metering Process Reliability Statistics Options Template, also specified in [I-D.ietf-ipfix-protocol-rfc5101bis], Mediators SHOULD implement the Intermediate Process Reliability Statistics Options Template, specified in the subsection below.

The Flow Keys Options Template, as specified in [I-D.ietf-ipfix-protocol-rfc5101bis], may require special handling at an IPFIX Mediator as described below.

In addition, each Intermediate Process may have its own specific reporting requirements (e.g. Anonymization Records as in [RFC6235], or the Aggregation Counter Distribution Options Template as in [I-D.ietf-ipfix-a9n]); these SHOULD be implemented as necessary as described in the specification for each Intermediate Process.

10.1. Intermediate Process Reliability Statistics Template

The Intermediate Process Statistics Options Template specifies the structure of a Data Record for reporting Intermediate Process statistics. It SHOULD contain the following Information Elements; the intermediateProcessId Information Element is defined in Section 10.3, and the ignoredRecordTotalCount Information Element is defined in Section 10.4:

IE	Description
observationDomainId [scope]	An identifier of the Observation Domain (of messages exported by this Mediator), locally unique to the Intermediate Process, to which this statistics record applies.
intermediateProcessId [scope]	An identifier for the Intermediate Process to which this statistics record applies.

ignoredRecordTotalCount	The total number of Data Records received but not processed by the Intermediate Process.
time first record ignored	The timestamp of the first record that was ignored by the Intermediate Process. For Data Records containing timestamp ranges, this SHOULD be taken from the start timestamp of the range; for data records containing no timing information, this SHOULD be taken from the Export Time in the message header of the containing IPFIX Message. For this timestamp, any of the following timestamp can be used: observationTimeSeconds, observationTimeMilliseconds, observationTimeMicroseconds, or observationTimeNanoseconds.
time last record ignored	The timestamp of the last record that was ignored by the Intermediate Process. For Data Records containing timestamp ranges, this SHOULD be taken from the end timestamp of the range; for data records containing no timing information, this SHOULD be taken from the Export Time in the message header of the containing IPFIX Message. For this timestamp, any of the following timestamp can be used: observationTimeSeconds, observationTimeMilliseconds, observationTimeMicroseconds, or observationTimeNanoseconds.

10.2. Flow Key Options Template

The Flow Keys Option Template specifies the structure of a Data Record for reporting the Flow Keys of reported Flows. A Flow Keys Data Record extends a particular Template Record that is referenced by its `templateId` identifier. The Template Record is extended by specifying which of the Information Elements contained in the corresponding Data Records describe Flow properties that serve as Flow Keys of the reported Flow. This Options Template is defined in section 4.4 of [I-D.ietf-ipfix-protocol-rfc5101bis], and SHOULD be used by Mediators for export as defined there.

When an Intermediate Process exports Data Records containing

different Flow Keys from those received from the Original Exporter, and the Original Exporter sent a Flow Keys Options record to the IPFIX Mediator, the IPFIX Mediator MUST export a Flow Keys Options record defining the the new set of Flow Keys.

10.3. intermediateProcessId Information Element

Description: An identifier of an Intermediate Process that is unique per IPFIX Device. Typically, this Information Element is used for limiting the scope of other Information Elements. Note that process identifiers may be assigned dynamically; ie., and Intermediate Process may be re-started with a different ID.

Data Type: unsigned32

Data Type Semantics: identifier

ElementId: TBD4

10.4. ignoredRecordTotalCount Information Element

Description: The total number of received Data Records that the Intermediate Process did not process since the (re-)initialization of the Intermediate Process; includes only Data Records not examined or otherwise handled by the Intermediate Process due to resource constraints, not Data Records which were examined or otherwise handled by the Intermediate Process but which merely do not contribute to any exported Data Record due to the operations performed by the Intermediate Process.

Data Type: unsigned64

Data Type Semantics: totalCounter

ElementId: TBD5

11. Configuration Management

In general, using IPFIX Mediators to combine information from multiple Original Exporters requires a consistent configuration of the Metering Processes behind these Original Exporters. The details of this consistency are specific to each Intermediate Process. Consistency of configuration should be verified out of band, with the MIB modules ([RFC6615] and [RFC6727]) or with the Configuration Data Model for IPFIX and PSAMP [RFC6728]

12. Security Considerations

As they act as both IPFIX Collecting Processes and Exporting Processes, the Security Considerations for IPFIX Protocol [I-D.ietf-ipfix-protocol-rfc5101bis] also apply to IPFIX Mediators. The Security Considerations for IPFIX Files [RFC5655] also apply to IPFIX Mediators that write IPFIX Files or use them for internal storage. However, there are a few specific considerations that IPFIX Mediator implementations must also take into account.

By design, IPFIX Mediators are "men-in-the-middle": they intercede in the communication between an Original Exporter (or another upstream IPFIX 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 an IPFIX Mediator. These are addressed in more detail in the Security Considerations for IPFIX Mediators in [RFC6183].

Note that, while IPFIX Mediators can use the exporterCertificate and collectorCertificate Information Elements defined in [RFC5655] as described in section 9.3 of [RFC6183] to export information about X.509 identities in upstream TLS-protected Transport Sessions, this mechanism cannot be used to provide true end-to-end assertions about a chain of IPFIX Mediators: any IPFIX Mediator in the chain can simply falsify the information about upstream Transport Sessions. In situations where information about the chain of mediation is important, it must be determined out of band.

13. IANA Considerations

This document specifies n new IPFIX Information Elements, originalExporterIPv4Address in Section 5.1, originalExporterIPv6Address in Section 5.2, and originalObservationDomainId in Section 6.1, to be added to the IPFIX Information Element registry [iana-ipfix-assignments]. [IANA NOTE: please add the three Information Elements as specified in the references subsections, and change TBD1, TBD2, and TBD3 in this document to reflect the assigned identifiers.]

14. Acknowledgments

We would like to thank the IPFIX contributors, specifically Paul Aitken for his thorough review and Rahul Patel for his feedback and comments. This work is materially supported by the European Union Seventh Framework Programme under grant agreement 257315 (DEMONS).

15. References

15.1. Normative References

- [RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.
- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3758] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", RFC 3758, May 2004.
- [RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.
- [RFC5655] Trammell, B., Boschi, E., Mark, L., Zseby, T., and A. Wagner, "Specification of the IP Flow Information Export (IPFIX) File Format", RFC 5655, October 2009.
- [RFC6313] Claise, B., Dhandapani, G., Aitken, P., and S. Yates, "Export of Structured Data in IP Flow Information Export (IPFIX)", RFC 6313, July 2011.
- [RFC6615] Dietz, T., Kobayashi, A., Claise, B., and G. Muenz, "Definitions of Managed Objects for IP Flow Information Export", RFC 6615, June 2012.
- [RFC6727] Dietz, T., Claise, B., and J. Quittek, "Definitions of Managed Objects for Packet Sampling", RFC 6727, October 2012.
- [RFC6728] Muenz, G., Claise, B., and P. Aitken, "Configuration Data Model for the IP Flow Information Export (IPFIX) and Packet Sampling (PSAMP) Protocols", RFC 6728, October 2012.
- [I-D.ietf-ipfix-protocol-rfc5101bis]
Claise, B. and B. Trammell, "Specification of the IP Flow Information eXport (IPFIX) Protocol for the Exchange of Flow Information", draft-ietf-ipfix-protocol-rfc5101bis-06 (work in progress), February 2013.
- [I-D.ietf-ipfix-information-model-rfc5102bis]

Claise, B. and B. Trammell, "Information Model for IP Flow Information eXport (IPFIX)",
draft-ietf-ipfix-information-model-rfc5102bis-10 (work in progress), February 2013.

[I-D.ietf-ipfix-flow-selection-tech]
D'Antonio, S., Zseby, T., Henke, C., and L. Peluso, "Flow Selection Techniques",
draft-ietf-ipfix-flow-selection-tech-13 (work in progress), February 2013.

[I-D.ietf-ipfix-a9n]
Trammell, B., Wagner, A., and B. Claise, "Flow Aggregation for the IP Flow Information Export (IPFIX) Protocol",
draft-ietf-ipfix-a9n-08 (work in progress), November 2012.

15.2. Informative References

- [RFC3917] Quittek, J., Zseby, T., Claise, B., and S. Zander, "Requirements for IP Flow Information Export (IPFIX)", RFC 3917, October 2004.
- [RFC3954] Claise, B., "Cisco Systems NetFlow Services Export Version 9", RFC 3954, October 2004.
- [RFC5470] Sadasivan, G., Brownlee, N., Claise, B., and J. Quittek, "Architecture for IP Flow Information Export", RFC 5470, March 2009.
- [RFC5472] Zseby, T., Boschi, E., Brownlee, N., and B. Claise, "IP Flow Information Export (IPFIX) Applicability", RFC 5472, March 2009.
- [RFC5476] Claise, B., Johnson, A., and J. Quittek, "Packet Sampling (PSAMP) Protocol Specifications", RFC 5476, March 2009.
- [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.
- [RFC5982] Kobayashi, A. and B. Claise, "IP Flow Information Export (IPFIX) Mediation: Problem Statement", RFC 5982, August 2010.
- [RFC6183] Kobayashi, A., Claise, B., Muenz, G., and K. Ishibashi, "IP Flow Information Export (IPFIX) Mediation: Framework", RFC 6183, April 2011.

[RFC6235] Boschi, E. and B. Trammell, "IP Flow Anonymization Support", RFC 6235, May 2011.

[iana-ipfix-assignments]
Internet Assigned Numbers Authority, "IP Flow Information
Export Information Elements
(<http://www.iana.org/assignments/ipfix/ipfix.xml>)".

[POSIX.1] IEEE, "IEEE 1003.1-2008 - IEEE Standard for Information
Technology - Portable Operating System Interface".

Authors' Addresses

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

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

Atsushi Kobayashi
NTT Information Sharing Platform Laboratories
3-9-11 Midori-cho
Musashino-shi, Tokyo 180-8585
Japan

Phone: +81 422 59 3978
Email: akoba@nttv6.net

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

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

Network Working Group
Internet Draft
Obsoletes: 5101
Category: Standards Track
Expires: August 22, 2013

B. Claise, Ed.
Cisco Systems, Inc.
B. Trammell, Ed.
ETH Zurich
February 18, 2013

Specification of the IP Flow Information eXport (IPFIX) Protocol
for the Exchange of Flow Information
draft-ietf-ipfix-protocol-rfc5101bis-06

Abstract

This document specifies the IP Flow Information Export (IPFIX) protocol that serves for transmitting Traffic Flow information over the network. In order to transmit Traffic Flow information from an Exporting Process to a Collecting Process, a common representation of flow data and a standard means of communicating them is required. This document describes how the IPFIX Data and Template Records are carried over a number of transport protocols from an IPFIX Exporting Process to an IPFIX Collecting Process. This document obsoletes RFC 5101.

Status of This Memo

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

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

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

This Internet-Draft will expire on August 22, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

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

Table of Contents

1. Introduction	4
1.1. Changes since RFC 5101	5
1.2. IPFIX Documents Overview	5
2. Terminology	7
2.1. Terminology Summary Table	12
3. IPFIX Message Format	12
3.1. Message Header Format	14
3.2. Field Specifier Format	15
3.3. Set and Set Header Format	16
3.3.1. Set Format	16
3.3.2. Set Header Format	17
3.4. Record Format	18
3.4.1. Template Record Format	18
3.4.2. Options Template Record Format	21
3.4.2.1. Scope	21
3.4.2.2. Options Template Record Format	21
3.4.3. Data Record Format	24
4. Specific Reporting Requirements	25
4.1. The Metering Process Statistics Options Template	25
4.2. The Metering Process Reliability Statistics Options Template	26
4.3. The Exporting Process Reliability Statistics Options Template	27
4.4. The Flow Keys Options Template	29
5. Timing Considerations	29
5.1 IPFIX Message Header Export Time and Flow Record Time	29
5.2 Supporting Timestamp Wraparound	30
6. Linkage with the Information Model	30
6.1. Encoding of IPFIX Data Types	31
6.1.1. Integral Data Types	31
6.1.2. Address Types	31
6.1.3. float32	31
6.1.4. float64	31
6.1.5. boolean	31
6.1.6. string and octetArray	31

6.1.7. dateTimeSeconds	32
6.1.8. dateTimeMilliseconds	32
6.1.9. dateTimeMicroseconds	32
6.1.10. dateTimeNanoseconds	32
6.2. Reduced Size Encoding	33
7. Variable-Length Information Element	34
8. Template Management	35
8.1. Template Withdrawal and Redefinition	36
8.2. Sequencing Template Management Actions	39
8.3. Additional considerations for Template Management over SCTP	39
8.4. Additional considerations for Template Management over UDP	40
9. The Collecting Process's Side	41
9.1. Additional considerations for SCTP Collecting Processes	42
9.2. Additional considerations for UDP Collecting Processes	42
10. Transport Protocol	43
10.1. Transport Compliance and Transport Usage	43
10.2. SCTP	44
10.2.1. Congestion Avoidance	44
10.2.2. Reliability	44
10.2.3. MTU	44
10.2.4. Association Establishment and Shutdown	44
10.2.5. Failover	45
10.2.6. Streams	45
10.3. UDP	46
10.3.1. Congestion Avoidance	46
10.3.2. Reliability	46
10.3.3. MTU	46
10.3.4. Session Establishment and Shutdown	46
10.3.5. Failover and Session Duplication	47
10.4. TCP	47
10.4.1. Congestion Avoidance	47
10.4.2. Reliability	47
10.4.3. MTU	48
10.4.4. Connection Establishment and Shutdown	48
10.4.5. Failover	48
11. Security Considerations	49
11.1. Applicability of TLS and DTLS	50
11.2. Usage	51
11.3. Mutual Authentication	51
11.4. Protection against DoS Attacks	52
11.5. When DTLS or TLS Is Not an Option	53
11.6. Logging an IPFIX Attack	53
11.7. Securing the Collector	54
12. IANA Considerations	54
Appendix A. IPFIX Encoding Examples	56
A.1. Message Header Example	56

A.2. Template Set Examples	57
A.2.1. Template Set Using IANA Information Elements	57
A.2.2. Template Set Using Enterprise-Specific Information Elements	57
A.3. Data Set Example	59
A.4. Options Template Set Examples	60
A.4.1. Options Template Set Using IANA Information Elements .	60
A.4.2. Options Template Set Using Enterprise-Specific Information	60
A.4.3. Options Template Set Using an Enterprise-Specific Scope	61
A.4.4. Data Set Using an Enterprise-Specific Scope	62
A.5. Variable-Length Information Element Examples	63
A.5.1. Example of Variable-Length Information Element with Length	63
A.5.2. Example of Variable-Length Information Element with 3 Octet Length Encoding	63
References	64
Normative References	64
Informative References	65
Acknowledgments	68
Authors' Addresses	68
Contributors' Addresses	69

1. Introduction

Traffic on a data network can be seen as consisting of flows passing through network elements. It is often interesting, useful, or even necessary to have access to information about these flows that pass through the network elements for administrative or other purposes. A Collecting Process should be able to receive the flow information passing through multiple network elements within the data network. This requires uniformity in the method of representing the flow information and the means of communicating the flows from the network elements to the collection point. This document specifies a protocol to achieve these requirements. This document specifies in detail the representation of different flows, the additional data required for flow interpretation, packet format, transport mechanisms used, security concerns, etc.

1.1. Changes since RFC 5101

This document obsoletes the Proposed Standard revision of the IPFIX Protocol Specification [RFC5101]. The protocol specified by this document is interoperable with the protocol as specified in [RFC5101]. The following changes have been made to this document with respect to the previous document:

- All outstanding technical and editorial errata on [RFC5101] have been addressed.
- The encoding of the `dateTimeSeconds`, `dateTimeMilliseconds`, `dateTimeMicroseconds`, and `dateTimeNanoseconds` data types, and the related encoding of the IPFIX Message Header Export Time field, have been clarified, especially with respect to the epoch upon which the timestamp data types are based.
- A new Section 5.2 has been added to address wraparound of these timestamp data types after they overflow in 2032 - 2038 CE (common era).
- Clarifications on encoding, especially in Section 6: all IPFIX values are encoded big-endian.
- Template management in section 8 has been simplified and clarified: the specification has been relaxed with respect to [RFC5101], especially concerning potential failures in Template ID reuse. Additional corner cases in template management have been addressed. The new template management language is interoperable with that in [RFC5101] to the extent that the behavior was defined in the prior specification.
- Section 11.3 (Mutual Authentication) has been improved to refer to current practices in TLS mutual authentication; references to Punycode were removed as these are covered in [RFC6125].
- Editorial improvements, including structural changes to sections 8, 9, and 10 to improve readability. Behavior common to all transport protocols has been separated out, with exceptions per transport specifically noted. All template management language (on both Exporting and Collecting Processes) has been unified in section 8.

1.2. IPFIX Documents Overview

The IPFIX protocol provides network administrators with access to IP flow information. The architecture for the export of measured IP flow information out of an IPFIX Exporting Process to a Collecting Process is defined in [RFC5470], per the requirements defined in

[RFC3917]. This document specifies how IPFIX Data Records and Templates are carried via a number of transport protocols from IPFIX Exporting Processes to IPFIX Collecting Processes.

Four IPFIX optimizations/extensions are currently specified: a bandwidth saving method for the IPFIX protocol in [RFC5473], an efficient method for exporting bidirectional flows in [RFC5103], a method for the definition and export of complex data structures in [RFC6313], and the specification of the Protocol for IPFIX Mediations [IPFIX-MED-PROTO] based on the IPFIX Mediation Framework [RFC6183].

A "file-based transport" for IPFIX, which defines how IPFIX Messages can be stored in files for document-based workflows and for archival purposes, is given in [RFC5655].

IPFIX has a formal description of IPFIX Information Elements, their name, type and additional semantic information, as specified in [RFC5102bis]. The registry is maintained by IANA [IPFIX-IANA]. The inline export of the Information Element type information is specified in [RFC5610].

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

[RFC6615] specifies a MIB module for monitoring, and [RFC6728] specifies a data model for configuring and monitoring IPFIX and PSAMP compliant devices using the NETCONF protocol. [RFC6727] specifies the PSAMP MIB module as an extension of the IPFIX SELECTOR MIB module defined in [RFC6615].

In terms of development, [RFC5153] provides guidelines for the implementation and use of the IPFIX protocol, while [RFC5471] provides guidelines for testing. Finally, [RFC5472] describes what type of applications can use the IPFIX protocol and how they can use the information provided. It furthermore shows how the IPFIX framework relates to other architectures and frameworks.

2. Terminology

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

The definitions of the basic terms like Traffic Flow, Exporting Process, Collecting Process, Observation Points, etc. are semantically identical to those found in the IPFIX requirements document [RFC3917]. Some of the terms have been expanded for more clarity when defining the protocol. Additional terms required for the protocol have also been defined. Definitions in this document and in [RFC5470] are equivalent; definitions that are only relevant to the IPFIX protocol only appear here.

The terminology summary table in Section 2.1 gives a quick overview of the relationships among some of the different terms defined.

Observation Point

An Observation Point is a location in the network where packets can be observed. Examples include: a line to which a probe is attached, a shared medium, such as an Ethernet-based LAN, a single port of a router, or a set of interfaces (physical or logical) of a router.

Note that every Observation Point is associated with an Observation Domain (defined below), and that one Observation Point may be a superset of several other Observation Points. For example, one Observation Point can be an entire line card. That would be the superset of the individual Observation Points at the line card's interfaces.

Observation Domain

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

Traffic Flow or Flow

There are several definitions of the term 'flow' being used by the Internet community. Within the context of IPFIX we use the following definition:

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

1. one or more packet header fields (e.g., destination IP address), transport header fields (e.g., destination port number), or application header fields (e.g., RTP header fields [RFC3550]).
2. one or more characteristics of the packet itself (e.g., number of MPLS labels, etc...).
3. one or more of fields derived from packet treatment (e.g., next hop IP address, the output interface, etc...).

A packet is defined as belonging to a Flow if it completely satisfies all the defined properties of the Flow.

Note that the set of packets represented by a Flow may be empty; that is, a Flow may represent zero or more packets. As sampling is a packet treatment, this definition includes packets selected by a sampling mechanism.

Flow Key

Each of the fields that:

1. belong to the packet header (e.g., destination IP address), or
2. are a property of the packet itself (e.g., packet length), or
3. are derived from packet treatment (e.g., Autonomous System (AS) number),

and that are used to define a Flow are termed Flow Keys.

Flow Record

A Flow Record contains information about a specific Flow that was observed at an Observation Point. A Flow Record contains measured

properties of the Flow (e.g., the total number of bytes for all the Flow's packets) and usually contains characteristic properties of the Flow (e.g., source IP address).

Metering Process

The Metering Process generates Flow Records. Inputs to the process are packet headers, characteristics, and packet treatment observed at one or more Observation Points.

The Metering Process consists of a set of functions that includes packet header capturing, timestamping, sampling, classifying, and maintaining Flow Records.

The maintenance of Flow Records may include creating new records, updating existing ones, computing Flow statistics, deriving further Flow properties, detecting Flow expiration, passing Flow Records to the Exporting Process, and deleting Flow Records.

Exporting Process

The Exporting Process sends IPFIX Messages to one or more Collecting Processes. The Flow Records in the Messages are generated by one or more Metering Processes.

Exporter

A device that hosts one or more Exporting Processes is termed an Exporter.

IPFIX Device

An IPFIX Device hosts at least one Exporting Process. It may host further Exporting Processes and arbitrary numbers of Observation Points and Metering Processes.

Collecting Process

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

Collector

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

Template

A Template is an ordered sequence of <type, length> pairs used to completely specify the structure and semantics of a particular set of information that needs to be communicated from an IPFIX Device to a Collector. Each Template is uniquely identifiable by means of a Template ID.

IPFIX Message

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

Message Header

The Message Header is the first part of an IPFIX Message, which provides basic information about the message, such as the IPFIX version, length of the message, message sequence number, etc.

Template Record

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

Data Record

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

Options Template Record

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

Set

A Set is a collection of records that have a similar structure, prefixed by a header. In an IPFIX Message, zero or more Sets follow the Message Header. There are three different types of Sets: Template Set, Options Template Set, and Data Set.

Template Set

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

Options Template Set

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

Data Set

A Data Set is one or more Data Records, of the same type, that are grouped together in an IPFIX Message. Each Data Record is previously defined by a Template Record or an Options Template Record.

Information Element

An Information Element is a protocol and encoding-independent description of an attribute that may appear in an IPFIX Record. The base set of Information Elements making up the IPFIX information model [RFC5102bis] are described in the IANA IPFIX Information Element Registry [IPFIX-IANA]. The type associated with an Information Element indicates constraints on what it may contain and also determines the valid encoding mechanisms for use in IPFIX.

Transport Session

In Stream Control Transmission Protocol (SCTP), the transport session is known as the SCTP association, which is uniquely identified by the SCTP endpoints [RFC4960]; in TCP, the transport session is known as the TCP connection, which is uniquely identified by the combination of IP addresses and TCP ports used. In UDP, the transport session is known as the UDP session, which is uniquely identified by the combination of IP addresses and UDP ports used.

2.1. Terminology Summary Table

Set	contents	
	Template	Record
Data Set	/	Data Record(s)
Template Set	Template Record(s)	/
Options Template Set	Options Template Record(s)	/

Figure A: Terminology Summary Table

A Data Set is composed of Data Record(s). No Template Record is included. A Template Record or an Options Template Record defines the Data Record.

A Template Set contains only Template Record(s).

An Options Template Set contains only Options Template Record(s).

3. IPFIX Message Format

An IPFIX Message consists of a Message Header, followed by zero or more Sets. The Sets can be any of the possible three types: Data Set, Template Set, or Options Template Set.

The format of the IPFIX Message is shown in Figure B.

Message Header
Set
Set
...
Set

Figure B: IPFIX Message Format

Following are some examples of IPFIX Messages:

1. An IPFIX Message consisting of interleaved Template, Data, and Options Template Sets, shown in Figure C. Here, Template and Options Template Sets are transmitted "on demand", before the first Data Set they define the structure of.

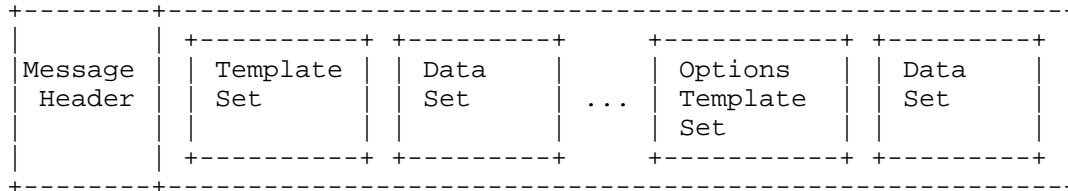


Figure C: IPFIX Message, Example 1

2. An IPFIX Message consisting entirely of Data Sets, sent after the appropriate Template Records have been defined and transmitted to the Collecting Process, shown in Figure D.

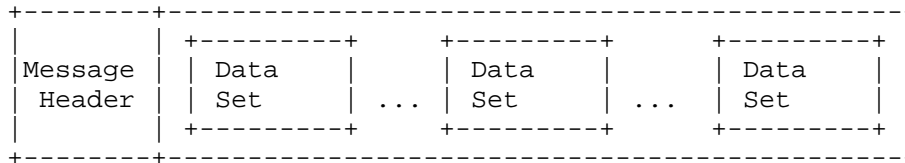


Figure D: IPFIX Message, Example 2

3. An IPFIX Message consisting entirely of Template and Options Template Sets, shown in Figure E. Such a message can be used to define or redefine Templates and Options Templates in bulk.

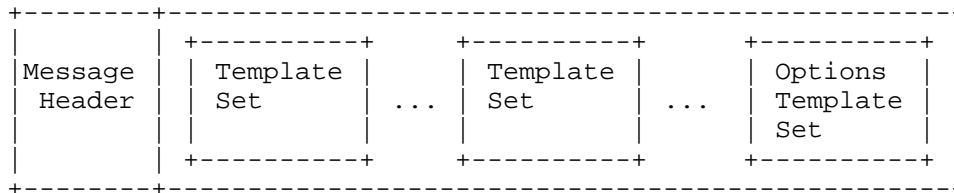


Figure E: IPFIX Message, Example 3

3.1. Message Header Format

The format of the IPFIX Message Header is shown in Figure F.

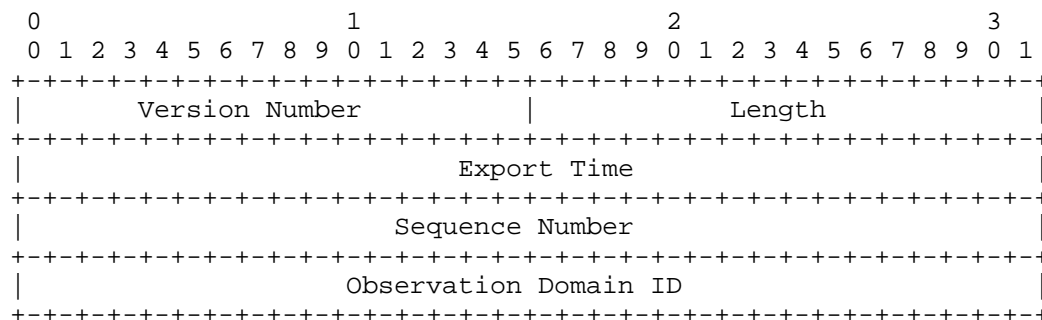


Figure F: IPFIX Message Header Format

Each Message Header field is exported in big-endian byte order. The fields are defined as follows:

Version

Version of IPFIX to which this Message conforms. The value of this field is 0x000a for the current version, incrementing by one the version used in the NetFlow services export version 9 [RFC3954].

Length

Total length of the IPFIX Message, measured in octets, including Message Header and Set(s).

Export Time

Time at which the IPFIX Message Header leaves the Exporter, expressed in seconds since the UNIX epoch of 1 January 1970 at 00:00 UTC, encoded as an unsigned 32-bit integer.

Sequence Number

Incremental sequence counter modulo 2^{32} of all IPFIX Data Records sent in the current stream from the current Observation Domain by the Exporting Process. Each SCTP Stream counts sequence numbers separately, while all messages in a TCP connection or UDP transport session are considered to be part of the same stream. This value SHOULD be used by the Collecting Process to identify whether any IPFIX Data Records have been missed. Template and Options Template Records do not increase the Sequence Number.

Observation Domain ID

A 32-bit identifier of the Observation Domain that is locally unique to the Exporting Process. The Exporting Process uses the Observation Domain ID to uniquely identify to the Collecting Process the Observation Domain that metered the Flows. It is RECOMMENDED that this identifier also be unique per IPFIX Device. Collecting Processes SHOULD use the Transport Session and the Observation Domain ID field to separate different export streams originating from the same Exporter. The Observation Domain ID SHOULD be 0 when no specific Observation Domain ID is relevant for the entire IPFIX Message, for example, when exporting the Exporting Process Statistics, or in case of a hierarchy of Collectors when aggregated Data Records are exported.

3.2. Field Specifier Format

Vendors need the ability to define proprietary Information Elements, because, for example, they are delivering a pre-standards product, or the Information Element is, in some way, commercially sensitive. This section describes the Field Specifier format for both IANA-registered Information Elements [IPFIX-IANA] and enterprise-specific Information Elements.

The Information Elements are identified by the Information Element identifier. When the Enterprise bit is set to 0, the corresponding Information Element appears in [IPFIX-IANA], and the Enterprise Number MUST NOT be present. When the Enterprise bit is set to 1, the corresponding Information Element identifier identified an enterprise-specific Information Element; the Enterprise Number MUST be present. An example of this is shown in Section A.2.2.

The Field Specifier format is shown in Figure G.

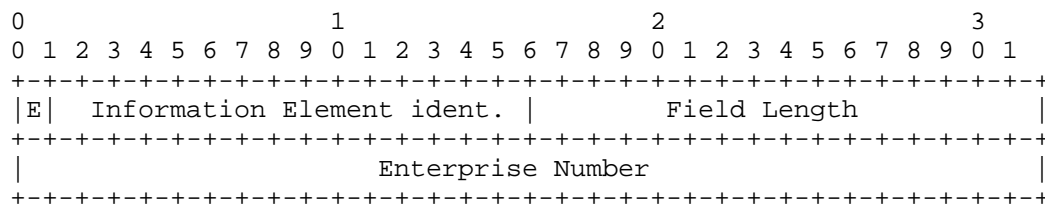


Figure G: Field Specifier Format

Where:

E

Enterprise bit. This is the first bit of the Field Specifier. If this bit is zero, the Information Element Identifier identifies an Information Element in [IPFIX-IANA], and the four-octet Enterprise Number field MUST NOT be present. If this bit is one, the Information Element identifier identifies an enterprise-specific Information Element, and the Enterprise Number field MUST be present.

Information Element identifier

A numeric value that represents the Information Element. Refer to [IPFIX-IANA].

Field Length

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

Enterprise Number

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

3.3. Set and Set Header Format

A Set is a generic term for a collection of records that have a similar structure. There are three different types of Sets: Template Sets, Options Template Sets, and Data Sets. Each of these Sets consists of a Set Header and one or more records. The Set Format and the Set Header Format are defined in the following sections.

3.3.1. Set Format

A Set has the format shown in Figure H. The record types can be either Template Records, Options Template Records, or Data Records. The record types MUST NOT be mixed within a Set.

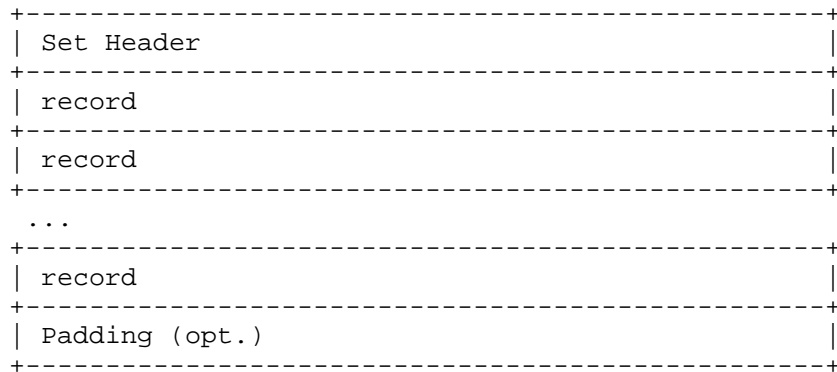


Figure H: Set Format

Set Header

The Set Header Format is defined in Section 3.3.2.

Record

One of the record Formats: Template Record, Options Template Record, or Data Record Format.

Padding

The Exporting Process MAY insert some padding octets, so that the subsequent Set starts at an aligned boundary. For security reasons, the padding octet(s) MUST be composed of zero (0) valued octets. The padding length MUST be shorter than any allowable record in this Set. If padding of the IPFIX Message is desired in combination with very short records, then the padding Information Element 'paddingOctets' can be used for padding records such that their length is increased to a multiple of 4 or 8 octets. Because Template Sets are always 4-octet aligned by definition, padding is only needed in case of other alignments e.g., on 8-octet boundaries.

3.3.2. Set Header Format

Every Set contains a common header. This header is defined in Figure I.

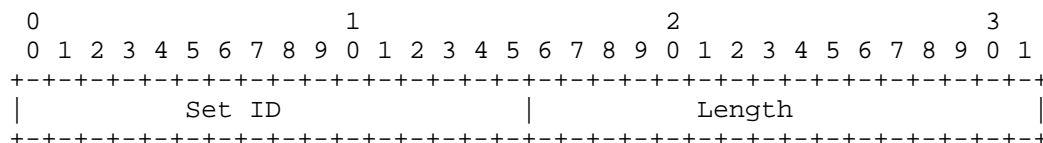


Figure I: Set Header Format

Each Set Header field is exported in big-endian format. The fields are defined as follows:

Set ID

Identifies the Set. A value of 2 is reserved for Template Sets. A value of 3 is reserved for Options Template Sets. Values from 4 to 255 are reserved for future use. Values 256 and above are used for Data Sets. The Set ID values of 0 and 1 are not used, for historical reasons [RFC3954].

Length

Total length of the Set, in octets, including the Set Header, all records, and the optional padding. Because an individual Set MAY contain multiple records, the Length value MUST be used to determine the position of the next Set.

3.4. Record Format

IPFIX defines three record formats, defined in the next sections: the Template Record Format, the Options Template Record Format, and the Data Record Format.

3.4.1. Template Record Format

One of the essential elements in the IPFIX record format is the Template Record. Templates greatly enhance the flexibility of the record format because they allow the Collecting Process to process IPFIX Messages without necessarily knowing the interpretation of all Data Records. A Template Record contains any combination of IANA-assigned and/or enterprise-specific Information Element identifiers.

The format of the Template Record is shown in Figure J. It consists of a Template Record Header and one or more Field Specifiers. The definition of the Field Specifiers is given in Figure G above.

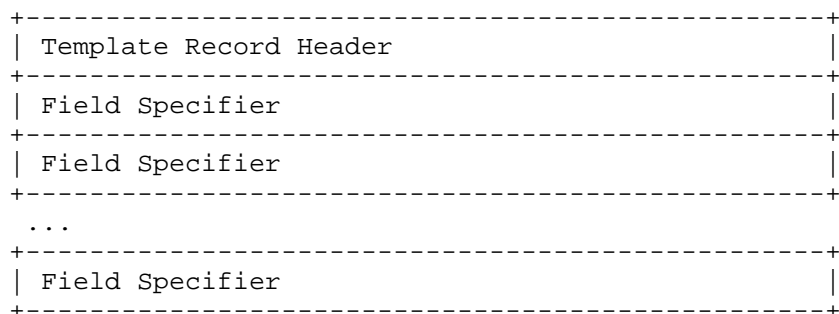


Figure J: Template Record Format

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

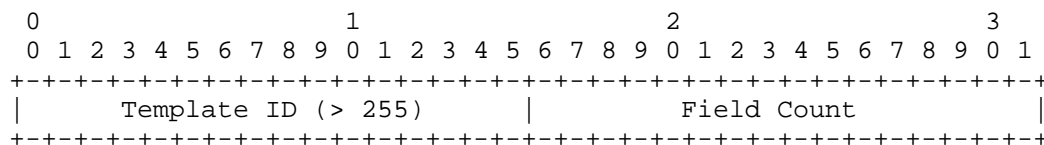


Figure K: Template Record Header Format

The Template Record Header Field Definitions are as follows:

Template ID

Each Template Record is given a unique Template ID in the range 256 to 65535. This uniqueness is local to the Transport Session and Observation Domain that generated the Template ID. Since Template IDs are used as Set IDs in the Sets they describe (see section 3.4.3), values 0-255 are reserved for special Set types (e.g. Template Sets themselves), and Templates and Options Templates (see section 3.4.2) cannot share Template IDs within a Transport Session and Observation Domain. There are no constraints regarding the order of the Template ID allocation. As Exporting Processes are free to allocate Template IDs as they see fit, Collecting Processes MUST NOT assume incremental Template IDs, or anything about the contents of a Template based on its Template ID alone.

Field Count

Number of fields in this Template Record.

The example in Figure L shows a Template Set with mixed IANA-assigned and enterprise-specific Information Elements. It consists of a Set Header, a Template Header, and several Field Specifiers.

0										1										2										3									
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1								
										Set ID = 2																				Length									
										Template ID = 256																				Field Count = N									
1										Information Element id. 1.1																				Field Length 1.1									
										Enterprise Number										1.1																			
0										Information Element id. 1.2																				Field Length 1.2									
																		
1										Information Element id. 1.N																				Field Length 1.N									
										Enterprise Number										1.N																			
										Template ID = 257																				Field Count = M									
0										Information Element id. 2.1																				Field Length 2.1									
1										Information Element id. 2.2																				Field Length 2.2									
										Enterprise Number										2.2																			
																		
1										Information Element id. 2.M																				Field Length 2.M									
										Enterprise Number										2.M																			
										Padding (opt)																													

Figure L: Template Set Example

Information Element Identifiers 1.2 and 2.1 appear in [IPFIX-IANA] (Enterprise bit = 0) and, therefore, do not need an Enterprise Number to identify them.

3.4.2. Options Template Record Format

Thanks to the notion of scope, The Options Template Record gives the Exporter the ability to provide additional information to the Collector that would not be possible with Flow Records alone.

See Section 4 for specific Options Templates used for reporting metadata about IPFIX Exporting and Metering Processes.

3.4.2.1. Scope

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

The scope is one or more Information Elements, specified in the Options Template Record. Collecting Processes SHOULD support as scope, at minimum, the observationDomainId, exportingProcessId, meteringProcessId, templateId, lineCardId, exporterIPv4Address, exporterIPv6Address, and ingressInterface Information Elements. The IPFIX protocol doesn't prevent the use of any Information Elements for scope. However, some Information Element types don't make sense if specified as scope; for example, the counter Information Elements.

The IPFIX Message Header already contains the Observation Domain ID. If not zero, this Observation Domain ID can be considered as an implicit scope for the Data Records in the IPFIX Message.

Multiple Scope Fields MAY be present in the Options Template Record, in which case, the composite scope is the combination of the scopes. For example, if the two scopes are meteringProcessId and templateId, the combined scope is this Template for this Metering Process. If a different order of Scope Fields would result in a Record having a different semantic meaning, then the order of Scope Fields MUST be preserved by the Exporting Process. For example, in the context of PSAMP [RFC5476], if the first scope defines the filtering function, while the second scope defines the sampling function, the order of the scope is important. Applying the sampling function first, followed by the filtering function, would lead to potentially different Data Records than applying the filtering function first, followed by the sampling function.

3.4.2.2. Options Template Record Format

An Options Template Record contains any combination of IANA-assigned and/or enterprise-specific Information Element identifiers.

The format of the Options Template Record is shown in Figure M. It consists of an Options Template Record Header and one or more Field Specifiers. The definition of the Field Specifiers is given in Figure G above.

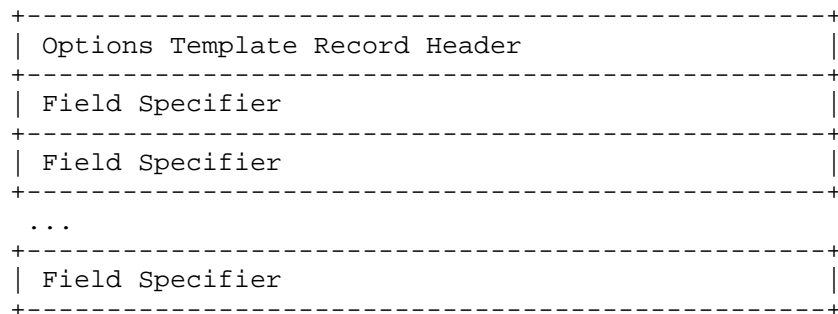


Figure M: Options Template Record Format

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

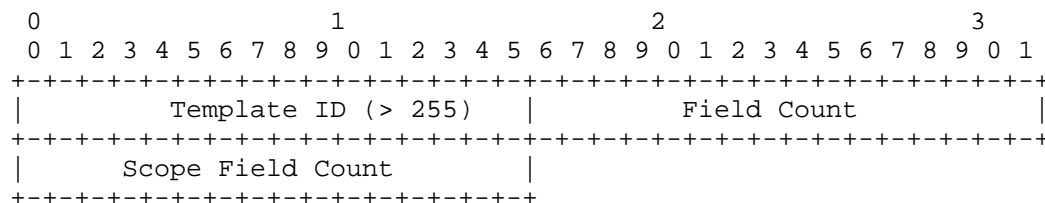


Figure N: Options Template Record Header Format

The Options Template Record Header Field Definitions are as follows:

Template ID

Each Options Template Record is given a unique Template ID in the range 256 to 65535. This uniqueness is local to the Transport Session and Observation Domain that generated the Template ID. Since Template IDs are used as Set IDs in the sets they describe (see section 3.4.3), values 0-255 are reserved for special Set types (e.g. Template Sets themselves), and Templates and Options Templates cannot share Template IDs within a Transport Session and Observation Domain. There are no constraints regarding the order of the Template ID allocation. As Exporting Processes are free to allocate Template IDs as they see fit, Collecting Processes MUST NOT assume incremental Template IDs, or anything about the contents of an Options Template based on its Template ID alone.

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. A scope field count of N specifies that the first N Field Specifiers in the Template Record are Scope Fields. The Scope Field Count MUST NOT be zero.

The example in Figure 0 shows an Options Template Set with mixed IANA-assigned and enterprise-specific Information Elements. It consists of a Set Header, a Options Template Header, and several Field Specifiers.

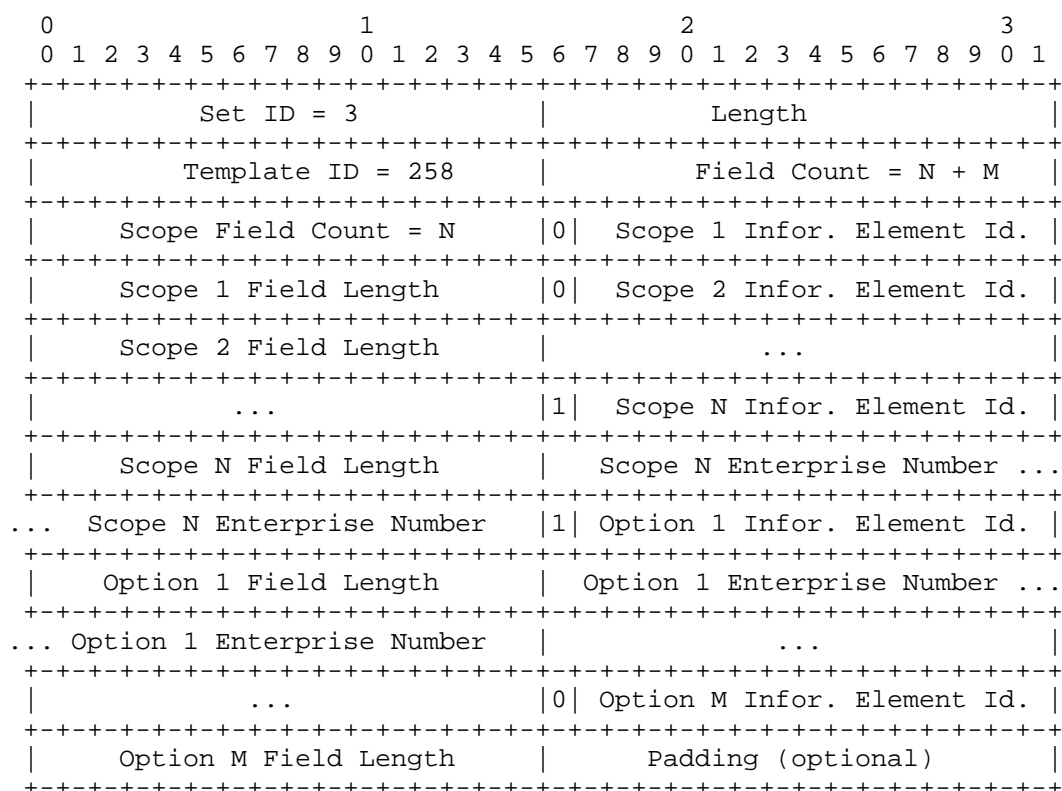


Figure 0: Options Template Set Example

3.4.3. Data Record Format

The Data Records are sent in Data Sets. The format of the Data Record is shown in Figure P. It consists only of one or more Field Values. The Template ID to which the Field Values belong is encoded in the Set Header field "Set ID", i.e., "Set ID" = "Template ID".

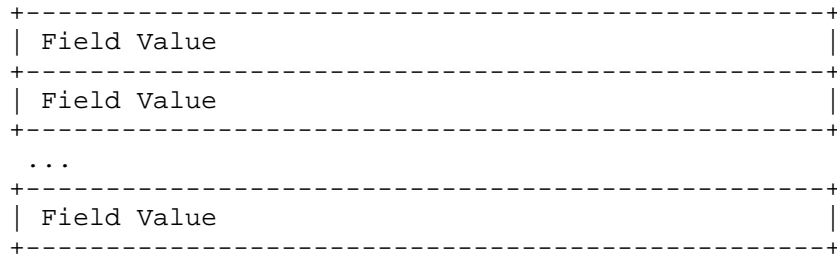


Figure P: Data Record Format

Note that Field Values do not necessarily have a length of 16 bits. Field Values are encoded according to their data type specified in [RFC5102bis].

Interpretation of the Data Record format can be done only if the Template Record corresponding to the Template ID is available at the Collecting Process.

The example in Figure Q shows a Data Set. It consists of a Set Header and several Field Values.

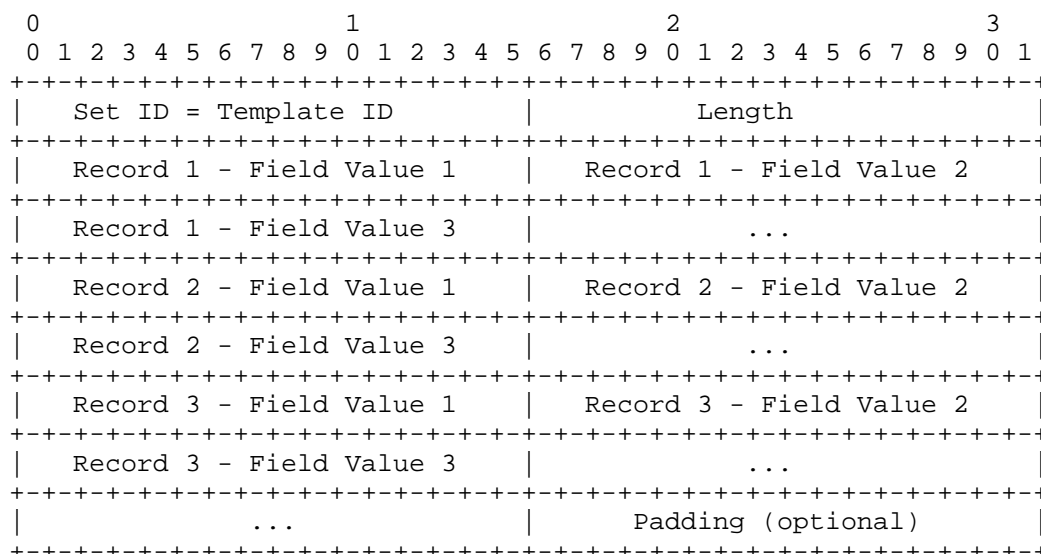


Figure Q: Data Set, containing Data Records

4. Specific Reporting Requirements

Some specific Options Templates and Options Template Records are necessary to provide extra information about the Flow Records and about the Metering Process.

The Options Template and Options Template Records defined in these subsections, which impose some constraints on the Metering Process and Exporting Process implementations, MAY be implemented. If implemented, the specific Options Templates SHOULD be implemented as specified in these subsections.

The minimum set of Information Elements is always specified in these Specific IPFIX Options Templates. Nevertheless, extra Information Elements may be used in these specific Options Templates.

The Collecting Process MUST check the possible combinations of Information Elements within the Options Template Records to correctly interpret the following Options Templates.

4.1. The Metering Process Statistics Options Template

The Metering Process Statistics Options Template specifies the structure of a Data Record for reporting Metering Process statistics. It SHOULD contain the following Information Elements; see [IPFIX-IANA] for their definitions

(scope) observationDomainId

This Information Element MUST be defined as a Scope Field, and MUST be present unless the Observation Domain ID of the enclosing Message is non-zero.

(scope) meteringProcessId

If present, this Information Element MUST be defined as a Scope Field.

exportedMessageTotalCount

exportedFlowRecordTotalCount

exportedOctetTotalCount

The Exporting Process SHOULD export the Data Record specified by the Metering Process Statistics Options Template on a regular basis or based on some export policy. This periodicity or export policy SHOULD be configurable.

Note that if several Metering Processes are available on the Exporter Observation Domain, the Information Element meteringProcessId MUST be specified as an additional Scope Field.

4.2. The Metering Process Reliability Statistics Options Template

The Metering Process Reliability Options Template specifies the structure of a Data Record for reporting lack of reliability in the Metering Process. It SHOULD contain the following Information Elements defined in [IPFIX-IANA]:

(scope) observationDomainId

This Information Element MUST be defined as a Scope Field, and MUST be present unless the Observation Domain ID of the enclosing Message is non-zero.

(scope) meteringProcessId

If present, this Information Element MUST be defined as a Scope Field.

ignoredPacketTotalCount

ignoredOctetTotalCount

time first packet ignored

The timestamp of the first packet that was ignored by the Metering Process. For this timestamp, any of the following timestamp Information Elements can be used:
observationTimeSeconds,
observationTimeMilliseconds,
observationTimeMicroseconds, or
observationTimeNanoseconds.

time last packet ignored

The timestamp of the last packet that was ignored by the Metering Process. For this timestamp, any of the following timestamp Information Elements can be used:
observationTimeSeconds,
observationTimeMilliseconds,
observationTimeMicroseconds, or
observationTimeNanoseconds.

The Exporting Process SHOULD export the Data Record specified by the Metering Process Reliability Statistics Options Template on a regular basis or based on some export policy. This periodicity or export policy SHOULD be configurable.

Note that if several Metering Processes are available on the Exporter Observation Domain, the Information Element meteringProcessId MUST be specified as an additional Scope Field.

Since the Metering Process Reliability Option Template contains two identical timestamp Information Elements, and since the order of the Information Elements in the Template Records is not guaranteed, the Collecting Process interprets the time interval of ignored packets as the range between the two values; see Section 5.2 for wraparound considerations.

4.3. The Exporting Process Reliability Statistics Options Template

The Exporting Process Reliability Options Template specifies the structure of a Data Record for reporting lack of reliability in the Exporting Process. It SHOULD contain the following Information Elements defined in [IPFIX-IANA]:

(scope) Exporting Process ID

The identifier of the Exporting Process for which reliability is reported. Any of the exporterIPv4Address, exporterIPv6Address, or exportingProcessId Information Elements can be used for this field. This Information Element MUST be defined as a Scope Field.

notSentFlowTotalCount

notSentPacketTotalCount

notSentOctetTotalCount

time first flow dropped

The time at which the first Flow Record was dropped by the Exporting Process. For this timestamp, any of the following timestamp can be used: observationTimeSeconds, observationTimeMilliseconds, observationTimeMicroseconds, or observationTimeNanoseconds.

time last flow dropped

The time at which the last Flow Record was dropped by the Exporting Process. For this timestamp, any of the following timestamp can be used: observationTimeSeconds, observationTimeMilliseconds, observationTimeMicroseconds, or observationTimeNanoseconds.

The Exporting Process SHOULD export the Data Record specified by the Exporting Process Reliability Statistics Options Template on a regular basis or based on some export policy. This periodicity or export policy SHOULD be configurable.

Since the Exporting Process Reliability Option Template contains two identical timestamp Information Elements, and since the order of the Information Elements in the Template Records is not guaranteed, the Collecting Process interprets the time interval of ignored packets as the range between the two values; see Section 5.2 for wraparound considerations.

4.4. The Flow Keys Options Template

The Flow Keys Options Template specifies the structure of a Data Record for reporting the Flow Keys of reported Flows. A Flow Keys Data Record extends a particular Template Record that is referenced by its templateId identifier. The Template Record is extended by specifying which of the Information Elements contained in the corresponding Data Records describe Flow properties that serve as Flow Keys of the reported Flow.

The Flow Keys Options Template SHOULD contain the following Information Elements that are defined in [IPFIX-IANA]:

(scope) templateId This Information Element MUST be defined as a Scope Field.

flowKeyIndicator

5. Timing Considerations

5.1 IPFIX Message Header Export Time and Flow Record Time

The IPFIX Message Header Export Time field is the time at which the IPFIX Message Header leaves the Exporter, using the same encoding as the dateTimeSeconds abstract data type [RFC5102bis], i.e., expressed in seconds since the UNIX epoch, 1 January 1970 at 00:00 UTC, encoded as an unsigned 32-bit integer.

Certain time-related Information Elements may be expressed as an offset from this Export Time. For example, Data Records requiring a microsecond precision can export the flow start and end times with the flowStartMicroseconds and flowEndMicroseconds Information Elements, which encode the absolute time in microseconds in terms of the NTP epoch, 1 January 1900 at 00:00 UTC, in a 64-bit field. An alternate solution is to export the flowStartDeltaMicroseconds and flowEndDeltaMicroseconds Information Elements in the Data Record, which respectively report the flow start and end time as negative offsets from the Export Time, as an unsigned 32-bit integer. This latter solution lowers the export bandwidth requirement, saving four bytes per timestamp, while increasing the load on the Exporter, as the Exporting Process must calculate the flowStartDeltaMicroseconds and flowEndDeltaMicroseconds of every single Data Record before exporting the IPFIX Message.

It must be noted that timestamps based on the Export Time impose some time constraints on the Data Records contained within the IPFIX Message. In the example of flowStartDeltaMicroseconds and flowEndDeltaMicroseconds Information Elements, the Data Record can

only contain records with timestamps within 71 minutes of the Export Time. Otherwise, the 32-bit counter would not be sufficient to contain the flow start time offset.

5.2 Supporting Timestamp Wraparound

The `dateTimeSeconds` abstract data type [RFC5102bis] and the Export Time Message Header field (Section 3.1) are encoded as 32-bit unsigned integers, expressed as seconds since the UNIX epoch, 1 January 1970 at 00:00 UTC, as defined in [POSIX.1]. These values will wrap around on 7 February 2106 at 06:28:16 UTC.

In order to support continued use of the IPFIX Protocol beyond this date, Exporting Processes SHOULD export `dateTimeSeconds` values and the Export Time Message Header field as the number of seconds since the UNIX epoch, 1 January 1970 at 00:00 UTC, modulo 2^{32} . Collecting Processes SHOULD use the current date, or other contextual information, to properly interpret `dateTimeSeconds` values and the Export Time Message Header field.

There are similar considerations for the NTP-based `dateTimeMicroseconds` and `dateTimeNanoseconds` abstract data types [RFC5102bis]. Exporting Processes SHOULD export `dateTimeMicroseconds` and `dateTimeNanoseconds` values as if the NTP Era [RFC5905] is implicit; Collecting Processes SHOULD use the current date, or other contextual information, to determine the NTP Era in order to properly interpret `dateTimeMicroseconds` and `dateTimeNanoseconds` values in received Data Records.

The `dateTimeMilliseconds` abstract data type will wrap around in approximately 500 billion years; the specification of the behavior of this abstract data type after that time is left as a subject of a future revision of this specification.

The long-term storage of files [RFC5655] for archival purposes is affected by timestamp wraparound, as the use of the current date to interpret timestamp values in files stored on the order of multiple decades in the past may lead to incorrect values; therefore, it is RECOMMENDED that such files be stored with contextual information to assist in the interpretation of these timestamps.

6. Linkage with the Information Model

As with values in the IPFIX Message Header and Set Header, values of all Information Elements [RFC5102bis], except for those of the string and `octetArray` data types, are encoded in canonical format in network byte order (also known as big-endian byte ordering).

6.1. Encoding of IPFIX Data Types

The following sections define the encoding of the data types specified in [RFC5102bis].

6.1.1. Integral Data Types

Integral data types -- `octet`, `signed8`, `unsigned16`, `signed16`, `unsigned32`, `signed32`, `signed64`, and `unsigned64` -- MUST be encoded using the default canonical format in network byte order. Signed Integral data types are represented in two's complement notation.

6.1.2. Address Types

Address types -- `macAddress`, `ipv4Address`, and `ipv6Address` -- MUST be encoded the same way as the integral data types, as six, four, and sixteen octets in network byte order, respectively.

6.1.3. float32

The float32 data type MUST be encoded as an IEEE single-precision 32-bit floating point-type, as specified in [IEEE.754.1985], in network byte order.

6.1.4. float64

The float64 data type MUST be encoded as an IEEE double-precision 64-bit floating point-type, as specified in [IEEE.754.1985], in network byte order.

6.1.5. boolean

The boolean data type is specified according to the `TruthValue` in [RFC2579]. It is encoded as a single-octet integer, as in Section 6.1.1., with the value 1 for true and a value 2 for false. Every other value is undefined.

6.1.6. string and octetArray

The data type `string` represents a finite length string of valid characters of the Unicode character encoding set. The `string` data type MUST be encoded in UTF-8 [RFC3629] format. The string is sent as an array of zero or more octets using an Information Element of fixed or variable length. IPFIX Exporting Processes MUST NOT send IPFIX Messages containing ill-formed UTF-8 string values for Information Elements of the `string` data type; Collecting Processes SHOULD detect and ignore such values. See [UTF8-EXPLOIT] for background on this issue.

The data type `octetArray` has no encoding rules; it represents a raw array of zero or more octets, with the interpretation of the octets defined in the Information Element definition.

6.1.7. `dateTimeSeconds`

The data type `dateTimeSeconds` is an unsigned 32-bit integer in network byte order containing the number of seconds since the UNIX epoch, 1 January 1970 at 00:00 UTC, as defined in [POSIX.1]. `dateTimeSeconds` is encoded identically to the IPFIX Message Header Export Time field. It can represent dates between 1 January 1970 and 7 February 2106 without wraparound; see section 5.2 for wraparound considerations.

6.1.8. `dateTimeMilliseconds`

The data type `dateTimeMilliseconds` is an unsigned 64-bit integer in network byte order, containing the number of milliseconds since the UNIX epoch, 1 January 1970 at 00:00 UTC, as defined in [POSIX.1]. It can represent dates beginning on 1 January 1970 for approximately the next 500 billion years without wraparound.

6.1.9 `dateTimeMicroseconds`

The data type `dateTimeMicroseconds` is a 64-bit field encoded according to the NTP Timestamp format as defined in section 6 of [RFC5905]. This field is made up of two unsigned 32-bit integers in network byte order, Seconds and Fraction. The Seconds field is the number of seconds since the NTP epoch, 1 January 1900 at 00:00 UTC. The Fraction field is the fractional number of seconds in units of $1/(2^{32})$ seconds (approximately 233 picoseconds). It can represent dates beginning between 1 January 1900 and 8 February 2036 in the current NTP Era; see section 5.2 for wraparound considerations.

Note that `dateTimeMicroseconds` and `dateTimeNanoseconds` share an identical encoding. The `dateTimeMicroseconds` data type is intended only to represent timestamps of microsecond precision. Therefore, the bottom 11 bits of the fraction field SHOULD be zero and MUST be ignored for all Information Elements of this data type (as $2^{11} \times 233$ picoseconds = .477 microseconds).

6.1.10 `dateTimeNanoseconds`

The data type `dateTimeNanoseconds` is a 64-bit field encoded according to the NTP Timestamp format as defined in section 6 of [RFC5905]. This field is made up of two unsigned 32-bit integers in network byte order, Seconds and Fraction. The Seconds field is the number of seconds since the NTP epoch, 1 January 1900 at 00:00 UTC. The

Fraction field is the fractional number of seconds in units of $1/(2^{32})$ seconds (approximately 233 picoseconds). It can represent dates beginning between 1 January 1900 and 8 February 2036 in the current NTP Era; see section 5.2 for wraparound considerations.

Note that `dateTimeMicroseconds` and `dateTimeNanoseconds` share an identical encoding. There is no restriction on the interpretation of the Fraction field for the `dateTimeNanoseconds` data type.

6.2. Reduced Size Encoding

Information Elements encoded as signed, unsigned, or float data types MAY be encoded using fewer octets than those implied by their type in the information model definition, based on the assumption that the smaller size is sufficient to carry any value the Exporter may need to deliver. This reduces the network bandwidth requirement between the Exporter and the Collector. Note that the Information Element definitions [IPFIX-IANA] always define the maximum encoding size.

For instance, the information model defines `octetDeltaCount` as an `unsigned64` type, which would require 64 bits. However, if the Exporter will never locally encounter the need to send a value larger than 4294967295, it may choose to send the value instead as an `unsigned32`.

This behavior is indicated by the Exporter by specifying a size in the Template with a smaller length than that associated with the assigned type of the Information Element. In the example above, the Exporter would place a length of 4 versus 8 in the Template.

Reduced size encoding MAY be applied to the following integer types: `unsigned64`, `signed64`, `unsigned32`, `signed32`, `unsigned16`, and `signed16`. The signed versus unsigned property of the reported value MUST be preserved. The reduction in size can be to any number of octets smaller than the original type if the data value still fits, i.e., so that only leading zeroes are dropped. For example, an `unsigned64` can be reduced in size to 7, 6, 5, 4, 3, 2, or 1 octet(s).

Reduced size encoding MAY be used to reduce `float64` to `float32`. The `float32` not only has a reduced number range, but due to the smaller mantissa, is also less precise. In this case, the `float64` would be reduced in size to 4 octets.

Reduced size encoding MUST NOT be applied to any other data type defined in [RFC5102bis] that implies a fixed length, as these types either have internal structure (such as `ipv4Address` or `dateTimeMicroseconds`) or restricted ranges that are not suitable for reduced length encoding (such as `dateTimeMilliseconds`).

Information Elements of type `octetArray` and `string` may be exported using any length, subject to restrictions on length specific to each Information Element, as noted in that Information Element's description.

7. Variable-Length Information Element

The IPFIX Template mechanism is optimized for fixed-length Information Elements [RFC5102bis]. Where an Information Element has a variable length, the following mechanism **MUST** be used to carry the length information for both the IANA and enterprise-specific Information Elements.

In the Template Set, the Information Element Field Length is recorded as 65535. This reserved length value notifies the Collecting Process that length of the Information Element will be carried in the Information Element content itself.

In most cases, the length of the Information Element will be less than 255 octets. The following length-encoding mechanism optimizes the overhead of carrying the Information Element length in this majority case. The length is carried in the octet before the Information Element, as shown in Figure R.

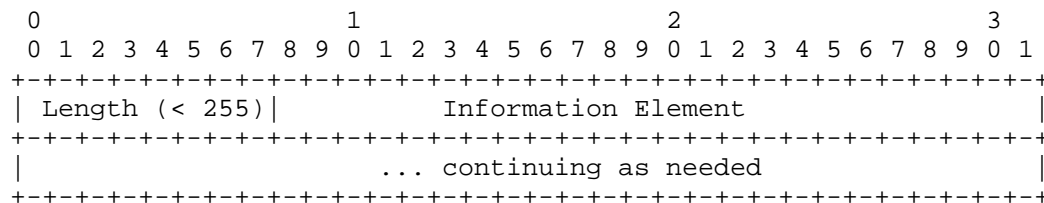


Figure R: Variable-Length Information Element (length < 255 octets)

The length may also be encoded into 3 octets before the Information element allowing the length of the Information Element to be greater than or equal to 255 octets. In this case, first octet of the Length field **MUST** be 255, and the length is carried in the second and third octets, as shown in Figure S.

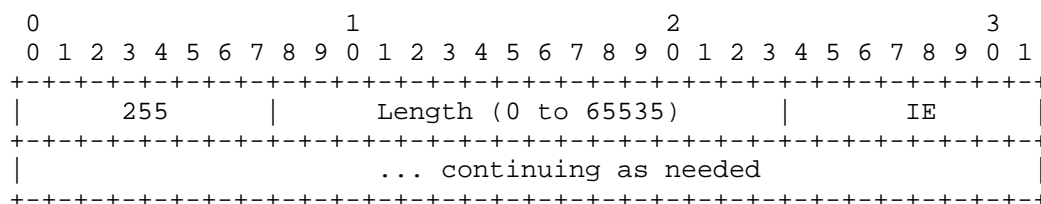


Figure S: Variable-Length Information Element (length 0 to 65535 octets)

The octets carrying the length (either the first or the first three octets) MUST NOT be included in the length of the Information Element.

8. Template Management

This section describes the management of Templates and Options Templates at the Exporting and Collecting Processes. The goal of Template management is to ensure, to the extent possible, that the Exporting Process and Collecting Process have a consistent view of the Templates and Options Templates used to encode and decode the Records sent from the Exporting Process to the Collecting Process. Achieving this goal is complicated somewhat by two factors: 1. the need to support the reuse of Template IDs within a Transport Session and 2. the need to support unreliable transmission for Templates when UDP is used as the transport protocol for IPFIX Messages.

The Template Management mechanisms defined in this section apply to IPFIX Messages export on SCTP, TCP, or UDP. Additional considerations specific to SCTP and UDP transport are given in sections 8.3 and 8.4, respectively.

The Exporting Process assigns and maintains Template IDs per Transport Session and Observation Domain. A newly created Template Record is assigned an unused Template ID by the Exporting Process. The Collecting Process MUST store all received Template Record information for the duration of each Transport Session until reuse or withdrawal as in section 8.1, or expiry over UDP as in section 8.4, so that it can interpret the corresponding Data Records.

The Collecting Process MUST NOT assume that the Template IDs from a given Exporting Process refer to the same Templates as they did in previous Transport Sessions from the same Exporting Process; a Collecting Process MUST NOT use Templates from one Transport Session to decode Data Sets in a subsequent Transport Session.

If a specific Information Element is required by a Template, but is not present in observed packets, the Exporting Process MAY choose to export Flow Records without this Information Element in a Data Record described by a new Template.

If an Information Element is required more than once in a Template, the different occurrences of this Information Element SHOULD follow the logical order of their treatments by the Metering Process. For example, if a selected packet goes through two hash functions, and if the two hash values are sent within a single Template, the first occurrence of the hash value should belong to the first hash function in the Metering Process. For example, when exporting the two source IP addresses of an IPv4-in-IPv4 packet, the first sourceIPv4Address Information Element occurrence should be the IPv4 address of the outer header, while the second occurrence should be the address of the inner header. Collecting Processes MUST properly handle Templates with multiple identical Information Elements.

The Exporting Process SHOULD transmit the Template Set and Options Template Set in advance of any Data Sets that use that (Options) Template ID, to help ensure that the Collector has the Template Record before receiving the first Data Record. Data Records that correspond to a Template Record MAY appear in the same and/or subsequent IPFIX Message(s). However, a Collecting Process MUST NOT assume that the Data Set and the associated Template Set (or Options Template Set) are exported in the same IPFIX Message.

Though a Collecting Process normally receives Template Records from the Exporting Process before receiving Data Records, this is not always the case, e.g. in case of reordering or Collecting Process restart over UDP. In these cases, the Collecting Process MAY buffer Data Records for which it has no Templates to wait for Template Records describing them; however, note that in the presence of Template withdrawal and redefinition (Section 8.1) this may lead to incorrect interpretation of Data Records.

Different Observation Domains within a Transport Session MAY use the same Template ID value to refer to different Templates; Collecting Processes MUST properly handle this case.

Options Templates and Templates which are related or interdependent (e.g. by sharing common properties as in [RFC5473]) SHOULD be sent together in the same IPFIX Message.

8.1. Template Withdrawal and Redefinition

Templates that will not be used further by an Exporting Process MAY be withdrawn by sending a Template Withdrawal. After receiving a

Template Withdrawal, a Collecting Process MUST stop using the Template to interpret subsequently-exported Data Sets. Note that this mechanism does not apply when UDP is used to transport IPFIX Messages; for this case, see Section 8.4.

A Template Withdrawal consists of a Template Record for the Template ID to be withdrawn, with a Field Count of 0. The format of a Template Withdrawal is shown in Figure T.

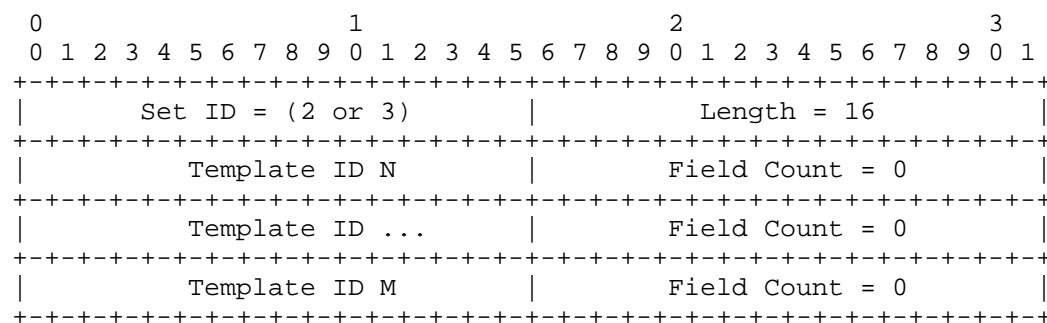


Figure T: Template Withdrawal Format

The Set ID field MUST contain the value 2 for Template Set Withdrawal and the value 3 for Options Template Set Withdrawal. Multiple Template IDs MAY be withdrawn with a single Template Withdrawal, in that case, padding MAY be used.

Template Withdrawals MAY appear interleaved with Template Sets, Options Template Sets, and Data Sets within an IPFIX Message. In this case, the Templates and Template Withdrawals shall be taken to take effect in the order in which they appear in the IPFIX Message. An Exporting Process SHOULD NOT send a Template Withdrawal until sufficient time has elapsed to allow receipt and processing of any Data Records described by the withdrawn Templates; see Section 8.2 on sequencing of Template management actions.

The end of a Transport Session implicitly withdraws all the Templates used within the Transport Session, and Templates must be resent during subsequent Transport Sessions between an Exporting Process and Collecting Process. This applies to SCTP and TCP only; see sections 8.4 and 10.3.4 for a discussion of Transport Session and Template lifetime over UDP.

All Templates for a given Observation Domain MAY also be withdrawn using an All Templates Withdrawal, shown in Figure U. All Options Templates for a given Observation Domain MAY likewise be withdrawn

using an All Options Templates Withdrawal, shown in Figure 3.

```

      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 = 8               |
+-----+-----+-----+-----+-----+-----+-----+-----+
|               Template ID = 2           |               Field Count = 0           |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Figure U: All Templates Withdrawal Set Format

```

      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 = 8               |
+-----+-----+-----+-----+-----+-----+-----+-----+
|               Template ID = 3           |               Field Count = 0           |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Figure V: All Options Templates Withdrawal Set Format

Template IDs MAY be reused for new Templates by sending a new Template Record or Options Template Record for a given Template ID after withdrawing the Template.

If a Collecting Process receives a Template Withdrawal for a Template or Options Template it does not presently have stored, this indicates a malfunctioning or improperly-implemented Exporting Process. The continued receipt and interpretation of Data Records is still possible, but it MUST ignore the Template Withdrawal and SHOULD log the error.

If a Collecting Process receives a new Template Record or Options Template Record for an already-allocated Template ID, and that Template or Options Template is identical to the already-received Template or Options Template, it SHOULD log the retransmission; however, this is not an error condition, as it does not affect the interpretation of data records.

If a Collecting Process receives a new Template Record or Options Template Record for an already-allocated Template ID, and that Template or Options Template is different from the already-received Template or Options Template, this indicates a malfunctioning or improperly-implemented Exporting Process. The continued receipt and unambiguous interpretation of Data Records for this Template ID is no longer possible, the Collecting Process SHOULD log the error; further Collecting Process actions are out of scope of this specification.

8.2 Sequencing Template Management Actions

Since there is no guarantee of the ordering of exported IPFIX Messages across SCTP Streams or over UDP, an Exporting Process **MUST** sequence all Template management actions (i.e., Template Records defining new Templates and Template Withdrawals withdrawing them) using the Export Time field in the IPFIX Message Header.

An Exporting Process **MUST NOT** export a Data Set described by a new Template in an IPFIX Message with an Export Time before the Export Time of the IPFIX Message containing that Template. If a new Template and a Data Set described by it appear in the same IPFIX Message, the Template Set containing the Template **MUST** appear before the Data Set in the Message.

An Exporting Process **MUST NOT** export any Data Sets described by a withdrawn Template in IPFIX Messages with an Export Time after the Export Time of the IPFIX Message containing the Template Withdrawal withdrawing that Template.

Put another way, a Template describes Data Records contained in IPFIX Messages with an Export Time between the Export Time of the IPFIX Message containing the Template Record and either the Export Time of the IPFIX Message containing the Template Withdrawal withdrawing it or the end of the Transport Session, inclusive.

Even if sent in-order, IPFIX Messages containing Template management actions could arrive at the Collecting Process out-of-order, i.e. if sent via UDP or via different SCTP streams. Given this, Template Withdrawals and subsequent reuse of Template IDs can significantly complicate the problem of determining Template lifetimes at the Collecting Process. A Collecting Process **MAY** implement a buffer and use Export Time information to disambiguate the order of Template management actions. This buffer, if implemented, **SHOULD** be configurable to impart a delay on the order of the maximum reordering delay experienced at the Collecting Process. Note, in this case, that the Collecting Process' clock is irrelevant: it is only comparing the Export Times of Messages to each other.

8.3. Additional considerations for Template Management over SCTP

The specifications in this section apply only to SCTP; in case of contradiction with specifications in Sections 8 or 8.1, this section takes precedence.

Template Sets and Options Template Sets **MAY** be sent on any SCTP stream. Data Sets sent on a given SCTP stream **MAY** be represented by Template Records exported on any SCTP stream.

Template Sets and Options Template Sets MUST be sent reliably, using SCTP ordered delivery.

Template Withdrawals MAY be sent on any SCTP stream. Template Withdrawals MUST be sent reliably, using SCTP ordered delivery. Template IDs MAY be reused by sending a Template Withdrawal and/or a new Template Record on a different SCTP stream than the stream on which the original Template was sent.

Additional Template Management considerations are given in [RFC6526], which specifies an extension to explicitly link Templates with SCTP streams. In exchange for more restrictive rules on the assignment of Template Records to SCTP streams, this extension allows fast, reliable reuse of Template IDs and estimation of Data Record loss per Template.

8.4. Additional considerations for Template Management over UDP

The specifications in this section apply only to UDP; in case of contradiction with specifications in Sections 8 or 8.1, this section takes precedence.

Since UDP provides no method for reliable transmission of Templates, Exporting Processes using UDP as the Transport Protocol MUST periodically retransmit each active Template at regular intervals. The Template retransmission interval MUST be configurable, as via the `templateRefreshTimeout` and `optionsTemplateRefreshTimeout` defined in [RFC6728]. Default settings for these values are deployment- and application-specific.

Before exporting any Data Records described by a given Template Record or Options Template Record, especially in the case of Template ID reuse as in section 8.1, the Exporting Process SHOULD send multiple copies of the Template Record in separate IPFIX Message, in order to help ensure the Collecting Process has received it.

In order to minimize resource requirements for Templates which are no longer being used by the Exporting Process, the Collecting Process MAY associate a lifetime with each Template received in a UDP Transport Session. Templates not refreshed by the Exporting Process within the lifetime can then be discarded by the Collecting Process. The Template lifetime at the Collecting Process MAY be exposed by a configuration parameter, or MAY be derived from observation of the interval of periodic Template retransmissions from the Exporting Process. In this latter case, the Template lifetime SHOULD default to at least 3 times the observed retransmission rate.

Template Withdrawals (Section 8.1) MUST NOT be sent by Exporting

Processes exporting via UDP, and MUST be ignored by Collecting Processes collecting via UDP. Template IDs MAY be reused by Exporting Processes by exporting a new Template for the Template ID after waiting at least 3 times the retransmission rate. Note that Template ID reuse may lead to incorrect interpretation of Data Records if the retransmission and lifetime are not properly configured.

When a Collecting Process receives a new Template Record or Options Template Record via UDP for an already-allocated Template ID, and that Template or Options Template is identical to the already-received Template or Options Template, it SHOULD NOT log the retransmission, as this is the normal operation of Template refresh over UDP.

When a Collecting Process receives a new Template Record or Options Template Record for an already-allocated Template ID, and that Template or Options Template is different from the already-received Template or Options Template, the Collecting Process MUST replace the Template or Options Template for that Template ID with the newly-received Template or Options Template. This is the normal operation of Template ID reuse over UDP.

As Template IDs are unique per UDP session and per Observation Domain, at any given time, the Collecting Process SHOULD maintain the following for all the current Template Records and Options Template Records: <IPFIX Device, Exporter source UDP port, Collector IP address, Collector destination UDP port, Observation Domain ID, Template ID, Template Definition, Last Received>.

9. The Collecting Process's Side

This section describes the handling of the IPFIX Protocol at the Collecting Process common to all Transport Protocols. Additional considerations for SCTP and UDP are given in Sections 9.1 and 9.2 respectively. Template management at Collecting Processes is covered in Section 8.

The Collecting Process MUST listen for association requests / connections to start new Transport Sessions from the Exporting Process.

The Collecting Process MUST note the Information Element identifier of any Information Element that it does not understand and MAY discard that Information Element from received Data Records.

The Collecting Process MUST accept padding in Data Records and Template Records. The padding size is the Set Length minus the size of the Set Header (4 octets for the Set ID and the Set Length),

modulo the Record size deduced from the Template Record.

The IPFIX protocol has a Sequence Number field in the Export header that increases with the number of IPFIX Data Records in the IPFIX Message. A Collector can detect out-of-sequence, dropped, or duplicate IPFIX Messages by tracking the Sequence Number. A Collector SHOULD provide a logging mechanism for tracking out-of-sequence IPFIX Messages. Such out-of-sequence IPFIX Messages may be due to Exporter resource exhaustion where it cannot transmit messages at their creation rate, an Exporting Process reset, congestion on the network link between the Exporter and Collector, Collector resource exhaustion where it cannot process the IPFIX Messages at their arrival rate, out-of-order packet reception, duplicate packet reception, or an attacker injecting false messages.

If the Collecting Process receives a malformed IPFIX Message it MUST discard the IPFIX Message and SHOULD log the error. A malformed IPFIX Message is one that cannot be interpreted due to nonsensical length values (e.g., a variable length Information Element longer than its enclosing Set, a Set longer than its enclosing IPFIX Message, an IPFIX Message shorter than an IPFIX Message Header) or a reserved Version value (which may indicate a future version of IPFIX is being used for export, but practically occurs most often when non-IPFIX data is sent to an IPFIX Collecting Process). Note that non-zero Set padding does not constitute a malformed IPFIX Message.

9.1. Additional considerations for SCTP Collecting Processes

As an Exporting Process may request and support more than one stream per SCTP association, the Collecting Process MUST support the opening of multiple SCTP streams.

9.2. Additional considerations for UDP Collecting Processes

A Transport Session for IPFIX Messages transported over UDP is defined from the point of view of the Exporting Process, and roughly corresponds to the time during which a given Exporting Process sends IPFIX Messages over UDP to a given Collecting Process. Since this is difficult to detect at the Collecting Process, the Collecting Process MAY discard all Transport Session state after no IPFIX Messages are received from a given Exporting Process within a given Transport Session during a configurable idle timeout.

The Collecting Process SHOULD accept Data Records without the associated Template Record (or other definitions such as Common Properties) required to decode the Data Record. If the Template Records or other definitions have not been received at the time Data Records are received, the Collecting Process MAY store the Data

Records for a short period of time and decode them after the Template Records or other definitions are received, comparing Export Times of IPFIX Messages containing the Template Records with those containing the Data Records as in Section 8.2. Note that this mechanism may lead to incorrectly interpreted records in the presence of Template ID reuse or other identifiers with limited lifetimes.

10. Transport Protocol

The IPFIX Protocol Specification has been designed to be transport protocol independent. Note that the Exporter can export to multiple Collecting Processes using independent transport protocols.

The IPFIX Message Header 16-bit Length field limits the length of an IPFIX Message to 65535 octets, including the header. A Collecting Process MUST be able to handle IPFIX Message lengths of up to 65535 octets.

10.1. Transport Compliance and Transport Usage

SCTP [RFC4960] using the PR-SCTP extension specified in [RFC3758] MUST be implemented by all compliant implementations. UDP [UDP] MAY also be implemented by compliant implementations. TCP [TCP] MAY also be implemented by compliant implementations.

SCTP should be used in deployments where Exporters and Collectors are communicating over links that are susceptible to congestion. SCTP is capable of providing any required degree of reliability when used with the PR-SCTP extension.

TCP may be used in deployments where Exporters and Collectors communicate over links that are susceptible to congestion, but SCTP is preferred due to its ability to limit back pressure on Exporters and its message versus stream orientation.

UDP may be used, although it is not a congestion-aware protocol. However, in this case the IPFIX traffic between Exporter and Collector must be separately contained or provisioned to minimize the risk of congestion-related loss.

By default, the Collecting Process listens for connections on SCTP, TCP, and/or UDP port 4739. By default, the Collecting Process listens for secure connections on SCTP, TCP, and/or UDP port 4740 (refer to the Security Considerations section). By default, the Exporting Process attempts to connect to one of these ports. It MUST be possible to configure both the Exporting and Collecting Processes to use different ports than the default.

10.2. Sctp

This section describes how IPFIX is transported over Sctp [RFC4960] using the PR-Sctp [RFC3758] extension.

10.2.1. Congestion Avoidance

The Sctp transport protocol provides the required level of congestion avoidance by design.

Sctp detects congestion in the end-to-end path between the IPFIX Exporting Process and the IPFIX Collecting Process, and limits the transfer rate accordingly. When an IPFIX Exporting Process has records to export, but detects that transmission by Sctp is temporarily impossible, it can either wait until sending is possible again, or it can decide to drop the record. In the latter case, the dropped export data SHOULD be accounted for, so that the amount of dropped export data can be reported using the mechanism in Section 4.3.

10.2.2. Reliability

The Sctp transport protocol is by default reliable, but has the capability to deliver messages with partial reliability [RFC3758].

Using reliable Sctp messages for the IPFIX export is not in itself a guarantee that all Data Records will be delivered. If there is congestion on the link from the Exporting Process to the Collecting Process, or if a significant number of retransmissions are required, the send queues on the Exporting Process may fill up; the Exporting Process MAY either suspend, export, or discard the IPFIX Messages. If Data Records are discarded the IPFIX Sequence Numbers used for export MUST reflect the loss of data.

10.2.3. MTU

Sctp provides the required IPFIX Message fragmentation service based on path MTU discovery.

10.2.4. Association Establishment and Shutdown

The IPFIX Exporting Process initiates an Sctp association with the IPFIX Collecting Process. The Exporting Process MAY establish more than one association (connection "bundle" in Sctp terminology) to the Collecting Process.

An Exporting Process MAY support more than one active association to different Collecting Processes (including the case of different

Collecting Processes on the same host).

When an Exporting Process is shut down, it SHOULD shut down the SCTP association.

When a Collecting Process no longer wants to receive IPFIX Messages, it SHOULD shut down its end of the association. The Collecting Process SHOULD continue to receive and process IPFIX Messages until the Exporting Process has closed its end of the association.

When a Collecting Process detects that the SCTP association has been abnormally terminated, it MUST continue to listen for a new association establishment.

When an Exporting Process detects that the SCTP association to the Collecting Process is abnormally terminated, it SHOULD try to re-establish the association.

Association timeouts SHOULD be configurable.

10.2.5. Failover

If the Collecting Process does not acknowledge the attempt by the Exporting Process to establish an association, the Exporting Process should retry using the SCTP exponential backoff feature. The Exporter MAY log an alarm if the time to establish the association exceeds a specified threshold, configurable on the Exporter.

If Collecting Process failover is supported by the Exporting Process, a second SCTP association MAY be opened in advance.

10.2.6. Streams

An Exporting Process MAY request more than one SCTP stream per association. Each of these streams may be used for the transmission of IPFIX Messages containing Data Sets, Template Sets, and/or Options Template Sets.

Depending on the requirements of the application, the Exporting Process may send Data Sets with full or partial reliability, using ordered or out-of-order delivery, over any SCTP stream established during SCTP Association setup.

An IPFIX Exporting Process MAY use any PR-SCTP Service Definition as per Section 4 of the PR-SCTP [RFC3758] specification when using partial reliability to transmit IPFIX Messages containing only Data Sets.

However, Exporting Processes SHOULD mark such IPFIX Messages for retransmission for as long as resource or other constraints allow.

10.3. UDP

This section describes how IPFIX is transported over UDP [UDP].

10.3.1. Congestion Avoidance

UDP has no integral congestion-avoidance mechanism. Its use over congestion-sensitive network paths is therefore not recommended. UDP MAY be used in deployments where Exporters and Collectors always communicate over dedicated links that are not susceptible to congestion, i.e., links that are over-provisioned compared to the maximum export rate from the Exporters.

10.3.2. Reliability

UDP is not a reliable transport protocol, and cannot guarantee delivery of messages. IPFIX Messages sent from the Exporting Process to the Collecting Process using UDP may therefore be lost. UDP MUST NOT be used unless the application can tolerate some loss of IPFIX Messages.

The Collecting Process SHOULD deduce the loss and reordering of IPFIX Data Records by looking at the discontinuities in the IPFIX Sequence Number. In the case of UDP, the IPFIX Sequence Number contains the total number of IPFIX Data Records sent for the UDP Transport Session prior to the receipt of this IPFIX Message, modulo 2^{32} . A Collector SHOULD detect out-of-sequence, dropped, or duplicate IPFIX Messages by tracking the Sequence Number.

Exporting Processes exporting IPFIX Messages via UDP MUST include a valid UDP checksum [UDP] in UDP datagrams including IPFIX messages.

10.3.3. MTU

The maximum size of exported messages MUST be configured such that the total packet size does not exceed the path MTU. If the path MTU is unknown, a maximum packet size of 512 octets SHOULD be used.

10.3.4. Session Establishment and Shutdown

As UDP is a connectionless protocol, there is no real session establishment or shutdown for IPFIX over UDP. An Exporting Process starts sending IPFIX Messages to a Collecting Process at one point in time, and stops sending them at another point in time. This can lead to some complications in Template management, which are outlined in

Section 8.4 above.

10.3.5. Failover and Session Duplication

Because UDP is not a connection-oriented protocol, the Exporting Process is unable to determine from the transport protocol that the Collecting Process is no longer able to receive the IPFIX Messages. Therefore, it cannot invoke a failover mechanism. However, the Exporting Process MAY duplicate the IPFIX Message to several Collecting Processes.

10.4. TCP

This section describes how IPFIX is transported over TCP [TCP].

10.4.1. Congestion Avoidance

TCP controls the rate at which data can be sent from the Exporting Process to the Collecting Process, using a mechanism that takes into account both congestion in the network and the capabilities of the receiver.

Therefore, an IPFIX Exporting Process may not be able to send IPFIX Messages at the rate that the Metering Process generates them, either because of congestion in the network or because the Collecting Process cannot handle IPFIX Messages fast enough. As long as congestion is transient, the Exporting Process can buffer IPFIX Messages for transmission. But such buffering is necessarily limited, both because of resource limitations and because of timeliness requirements, so ongoing and/or severe congestion may lead to a situation where the Exporting Process is blocked.

When an Exporting Process has Data Records to export but the transmission buffer is full, and it wants to avoid blocking, it can decide to drop some Data Records. The dropped Data Records MUST be accounted for, so that the number of lost records can later be reported as in Section 4.3.

10.4.2. Reliability

TCP ensures reliable delivery of data from the Exporting Process to the Collecting Process.

10.4.3. MTU

As TCP offers a stream service instead of a datagram or sequential packet service, IPFIX Messages transported over TCP are instead separated using the Length field in the IPFIX Message Header. The Exporting Process can choose any valid length for exported IPFIX Messages, as TCP handles segmentation.

However, if an Exporting Process exports data from multiple Observation Domains, it should be careful to choose IPFIX Message lengths appropriately to minimize head-of-line blocking between different Observation Domains. Multiple TCP connections MAY be used to avoid head-of-line blocking between different Observation Domains.

10.4.4. Connection Establishment and Shutdown

The IPFIX Exporting Process initiates a TCP connection to the Collecting Process. An Exporting Process MAY support more than one active connection to different Collecting Processes (including the case of different Collecting Processes on the same host).

The Exporter MAY log an alarm if the time to establish the connection exceeds a specified threshold, configurable on the Exporter.

When an Exporting Process is shut down, it SHOULD shut down the TCP connection.

When a Collecting Process no longer wants to receive IPFIX Messages, it SHOULD close its end of the connection. The Collecting Process SHOULD continue to read IPFIX Messages until the Exporting Process has closed its end.

When a Collecting Process detects that the TCP connection to the Exporting Process has terminated abnormally, it MUST continue to listen for a new connection.

When an Exporting Process detects that the TCP connection to the Collecting Process has terminated abnormally, it SHOULD try to re-establish the connection. Connection timeouts and retry schedules SHOULD be configurable. In the default configuration, an Exporting Process MUST NOT attempt to establish a connection more frequently than once per minute.

10.4.5. Failover

If the Collecting Process does not acknowledge an attempt by the Exporting Process to establish a connection, TCP will automatically retry connection establishment using exponential backoff. The

Exporter MAY log an alarm if the time to establish the association exceeds a specified threshold, configurable on the Exporter.

If Collecting Process failover is supported by the Exporting Process, a second TCP connection MAY be opened in advance.

11. Security Considerations

The security considerations for the IPFIX protocol have been derived from an analysis of potential security threats, as discussed in the "Security Considerations" section of IPFIX requirements [RFC3917]. The requirements for IPFIX security are as follows:

1. IPFIX must provide a mechanism to ensure the confidentiality of IPFIX data transferred from an Exporting Process to a Collecting Process, in order to prevent disclosure of Flow Records transported via IPFIX.
2. IPFIX must provide a mechanism to ensure the integrity of IPFIX data transferred from an Exporting Process to a Collecting Process, in order to prevent the injection of incorrect data or control information (e.g., Templates) into an IPFIX Message stream.
3. IPFIX must provide a mechanism to authenticate IPFIX Collecting and Exporting Processes, to prevent the collection of data from an unauthorized Exporting Process or the export of data to an unauthorized Collecting Process.

Because IPFIX can be used to collect information for network forensics and billing purposes, attacks designed to confuse, disable, or take information from an IPFIX collection system may be seen as a prime objective during a sophisticated network attack.

An attacker in a position to inject false messages into an IPFIX Message stream can either affect the application using IPFIX (by falsifying data), or the IPFIX Collecting Process itself (by modifying or revoking Templates, or changing options); for this reason, IPFIX Message integrity is important.

The IPFIX Messages themselves may also contain information of value to an attacker, including information about the configuration of the network as well as end-user traffic and payload data, so care must be taken to confine their visibility to authorized users. When an Information Element containing end-user payload information is exported, it SHOULD be transmitted to the Collecting Process using a means that secures its contents against eavesdropping. Suitable mechanisms include the use of either a direct point-to-point

connection or the use of an encryption mechanism. It is the responsibility of the Collecting Process to provide a satisfactory degree of security for this collected data, including, if necessary, anonymization of any reported data.

11.1. Applicability of TLS and DTLS

Transport Layer Security (TLS) [RFC5246] and Datagram Transport Layer Security (DTLS) [RFC6347] were designed to provide the confidentiality, integrity, and authentication assurances required by the IPFIX protocol, without the need for pre-shared keys.

With the mandatory SCTP transport protocol for IPFIX, DTLS [RFC6347] MUST be implemented. If UDP is selected as the IPFIX transport protocol, DTLS [RFC6347] MUST be implemented. If TCP is selected as the IPFIX transport protocol, TLS [RFC5246] MUST be implemented.

Note that DTLS is selected as the security mechanism for SCTP. Though TLS bindings to SCTP are defined in [RFC3436], they require all communication to be over reliable, bidirectional streams, and require one TLS connection per stream. This arrangement is not compatible with the rationale behind the choice of SCTP as an IPFIX transport protocol.

Note that using DTLS [RFC6347] has a vulnerability, i.e., a true man in the middle may attempt to take data out of an association and fool the sender into thinking that the data was actually received by the peer. In generic TLS for SCTP (and/or TCP), this is not possible. This means that the removal of a message may become hidden from the sender or receiver. Another vulnerability of using SCTP with DTLS is that someone could inject SCTP control information to shut down the SCTP association, effectively generating a loss of IPFIX Messages if those are buffered outside of the SCTP association. Techniques such as [RFC6083] could be used to overcome these vulnerabilities.

When using DTLS over SCTP, the Exporting Process MUST ensure that each IPFIX Message is sent over the same SCTP stream that would be used when sending the same IPFIX Message directly over SCTP. Note that DTLS may send its own control messages on stream 0 with full reliability; however, this will not interfere with the processing of stream 0 IPFIX Messages at the Collecting Process, because DTLS consumes its own control messages before passing IPFIX Messages up to the application layer.

When using DTLS over SCTP or UDP, the Heartbeat Extension [RFC6520] SHOULD be used, especially on long-lived Transport Sessions, to ensure that the association remains active.

11.2. Usage

The IPFIX Exporting Process initiates the communication to the IPFIX Collecting Process, and acts as a TLS or DTLS client according to [RFC5246] and [RFC6347], while the IPFIX Collecting Process acts as a TLS or DTLS server. The DTLS client opens a secure connection on the SCTP port 4740 of the DTLS server if SCTP is selected as the transport protocol. The TLS client opens a secure connection on the TCP port 4740 of the TLS server if TCP is selected as the transport protocol. The DTLS client opens a secure connection on the UDP port 4740 of the DTLS server if UDP is selected as the transport protocol.

11.3. Mutual Authentication

When using TLS or DTLS, IPFIX Exporting Processes and IPFIX Collecting Processes SHOULD be identified by a certificate containing the DNS-ID identifier as in Section 6.4 of [RFC6125]; the inclusion of Common Names (CN-IDs) in certificates identifying IPFIX Exporting Processes or Collecting Processes is NOT RECOMMENDED.

To prevent man-in-the-middle attacks from impostor Exporting or Collecting Processes, the acceptance of data from an unauthorized Exporting Process, or the export of data to an unauthorized Collecting Process, mutual authentication MUST be used for both TLS and DTLS. Exporting Processes MUST verify the reference identifiers of the Collecting Processes they are exporting IPFIX messages to against those stored in the certificates. Likewise, Collecting Processes MUST verify the reference identifiers of the Exporting Processes they are receiving IPFIX Messages from against those stored in the certificates. Exporting Processes MUST NOT export to non-verified Collecting Processes, and Collecting Processes MUST NOT accept IPFIX Messages from non-verified Exporting Processes.

Exporting Processes and Collecting Processes MUST support the verification of certificates against an explicitly authorized list of peer certificates identified by Common Name, and SHOULD support the verification of reference identifiers by matching the DNS-ID or CN-ID with a DNS lookup of the peer.

IPFIX Exporting Processes and Collecting Processes MUST use non-NULL ciphersuites for authentication, integrity, and confidentiality. IPFIX Exporting Processes and Collecting Processes MUST support TLS version 1.1 and SHOULD support TLS version 1.2 [RFC5246]; Exporting and Collecting Processes MUST NOT request, offer, or use any version of SSL, or any version of TLS prior to 1.1, due to known security vulnerabilities in prior versions of the protocol; see Appendix E of [RFC5246] for more information.

11.4. Protection against DoS Attacks

An attacker may mount a denial-of-service (DoS) attack against an IPFIX collection system either directly, by sending large amounts of traffic to a Collecting Process, or indirectly, by generating large amounts of traffic to be measured by a Metering Process.

Direct DoS attacks can also involve state exhaustion, whether at the transport layer (e.g., by creating a large number of pending connections), or within the IPFIX Collecting Process itself (e.g., by sending Flow Records pending Template or scope information, a large amount of Options Template Records, etc.).

SCTP mandates a cookie-exchange mechanism designed to defend against SCTP state exhaustion DoS attacks. Similarly, TCP provides the "SYN cookie" mechanism to mitigate state exhaustion; SYN cookies SHOULD be used by any Collecting Process accepting TCP connections. DTLS also provides cookie exchange to protect against DTLS server state exhaustion.

The reader should note that there is no way to prevent fake IPFIX Message processing (and state creation) for UDP & SCTP communication. The use of TLS and DTLS can obviously prevent the creation of fake states, but they are themselves prone to state exhaustion attacks. Therefore, Collector rate limiting SHOULD be used to protect TLS & DTLS (like limiting the number of new TLS or DTLS session per second to a sensible number).

IPFIX state exhaustion attacks can be mitigated by limiting the rate at which new connections or associations will be opened by the Collecting Process, the rate at which IPFIX Messages will be accepted by the Collecting Process, and adaptively limiting the amount of state kept, particularly records waiting on Templates. These rate and state limits MAY be provided by a Collecting Process; if provided, the limits SHOULD be user configurable.

Additionally, an IPFIX Collecting Process can eliminate the risk of state exhaustion attacks from untrusted nodes by requiring TLS or DTLS mutual authentication, causing the Collecting Process to accept IPFIX Messages only from trusted sources.

With respect to indirect denial of service, the behavior of IPFIX under overload conditions depends on the transport protocol in use. For IPFIX over TCP, TCP congestion control would cause the flow of IPFIX Messages to back off and eventually stall, blinding the IPFIX system. SCTP improves upon this situation somewhat, as some IPFIX Messages would continue to be received by the Collecting Process due to the avoidance of head-of-line blocking by SCTP's multiple streams

and partial reliability features, possibly affording some visibility of the attack. The situation is similar with UDP, as some datagrams may continue to be received at the Collecting Process, effectively applying sampling to the IPFIX Message stream, implying that some forensics may be left.

To minimize IPFIX Message loss under overload conditions, some mechanism for service differentiation could be used to prioritize IPFIX traffic over other traffic on the same link. Alternatively, IPFIX Messages can be transported over a dedicated network. In this case, care must be taken to ensure that the dedicated network can handle the expected peak IPFIX Message traffic.

11.5. When DTLS or TLS Is Not an Option

The use of DTLS or TLS might not be possible in some cases due to performance issues or other operational concerns.

Without TLS or DTLS mutual authentication, IPFIX Exporting Processes and Collecting Processes can fall back on using IP source addresses to authenticate their peers. A policy of allocating Exporting Process and Collecting Process IP addresses from specified address ranges, and using ingress filtering to prevent spoofing, can improve the usefulness of this approach. Again, completely segregating IPFIX traffic on a dedicated network, where possible, can improve security even further. In any case, the use of open Collecting Processes (those that will accept IPFIX Messages from any Exporting Process regardless of IP address or identity) is discouraged.

Modern TCP and SCTP implementations are resistant to blind insertion attacks (see [RFC4960], [RFC6528]); however, UDP offers no such protection. For this reason, IPFIX Message traffic transported via UDP and not secured via DTLS SHOULD be protected via segregation to a dedicated network.

11.6. Logging an IPFIX Attack

IPFIX Collecting Processes MUST detect potential IPFIX Message insertion or loss conditions by tracking the IPFIX Sequence Number, and SHOULD provide a logging mechanism for reporting out-of-sequence messages. Note that an attacker may be able to exploit the handling of out-of-sequence messages at the Collecting Process, so care should be taken in handling these conditions. For example, a Collecting Process that simply resets the expected Sequence Number upon receipt of a later Sequence Number could be temporarily blinded by deliberate injection of later Sequence Numbers.

IPFIX Exporting and Collecting Processes SHOULD log any connection attempt that fails due to authentication failure, whether due to being presented an unauthorized or mismatched certificate during TLS or DTLS mutual authentication, or due to a connection attempt from an unauthorized IP address when TLS or DTLS is not in use.

IPFIX Exporting and Collecting Processes SHOULD detect and log any SCTP association reset or TCP connection reset.

11.7. Securing the Collector

The security of the Collector and its implementation is important to achieve overall security; however, a complete set of security guidelines for Collector implementation is outside the scope of this document.

As IPFIX uses length-prefix encodings, Collector implementors should take care to ensure detection of and proper operation despite inconsistent values that could impact IPFIX Message decoding. Specifically, IPFIX Message, Set, and variable-length Information Element lengths must be checked for consistency to avoid buffer-sizing vulnerabilities.

Collector implementors should also pay special attention to UTF-8 encoding of string datatypes, as vulnerabilities may exist in the interpretation of ill-formed UTF-8 values; see Section 6.1.6.

12. IANA Considerations

On publication of this document, IANA will update the IPFIX Information Element Registry [IPFIX-IANA] to update all references to RFC5101 to point to this document, instead.

This document has no further actions for IANA; the text below is explanatory.

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 Information Elements used by IPFIX, and sub-registries of Information Element values, are managed by IANA [IPFIX-IANA], as are the Private Enterprise Numbers used by enterprise-specific Information Elements [PEN-IANA]. This document makes no changes to these registries.

The IPFIX Version Number value of 0x000a (10) is reserved for the

IPFIX protocol specified in this document. Set ID values of 0 and 1 are not used, for historical reasons [RFC3954]. The Set ID value of 2 is reserved for the Template Set. The Set ID value of 3 is reserved for the Options Template Set. All other Set ID values from 4 to 255 are reserved for future use. Set ID values above 255 are used for Data Sets.

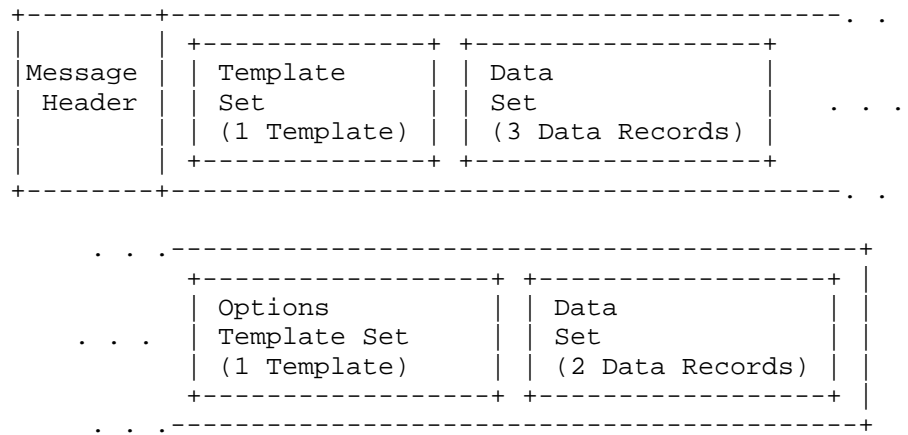
New assignments in either IPFIX Version Number or IPFIX Set ID assignments require a Standards Action [RFC5226], i.e., they are to be made via Standards Track RFCs approved by the IESG.

Appendix A. IPFIX Encoding Examples

This appendix, which is a not a normative reference, contains IPFIX encoding examples.

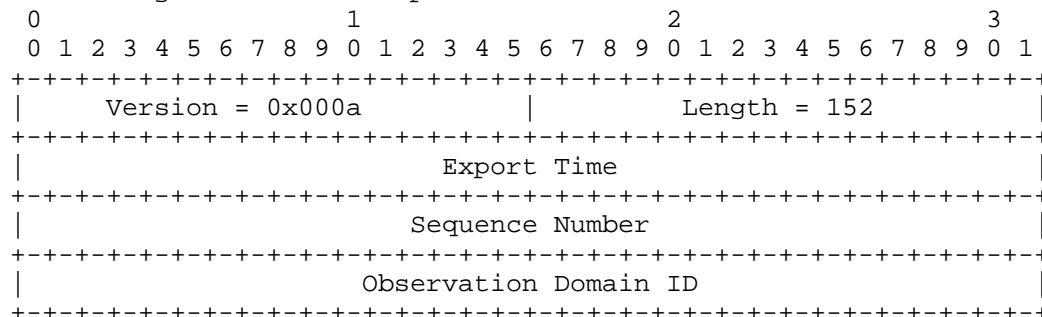
Let's consider the example of an IPFIX Message composed of a Template Set, a Data Set (which contains three Data Records), an Options Template Set and a Data Set (which contains 2 Data Records related to the previous Options Template Record).

IPFIX Message:



A.1. Message Header Example

The Message Header is composed of:



A.2. Template Set Examples

A.2.1. Template Set Using IANA Information Elements

We want to report the following Information Elements:

- The IPv4 source IP address: sourceIPv4Address in [IPFIX-IANA], with a length of 4 octets
- The IPv4 destination IP address: destinationIPv4Address in [IPFIX-IANA], with a length of 4 octets
- The next-hop IP address (IPv4): ipNextHopIPv4Address in [IPFIX-IANA], with a length of 4 octets
- The number of packets of the Flow: packetDeltaCount in [IPFIX-IANA], with a length of 4 octets
- The number of octets of the Flow: octetDeltaCount in [IPFIX-IANA], with a length of 4 octets

Therefore, the Template Set will be composed of the following:

0	1																2																3																
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9										
Set ID = 2																	Length = 28 octets																																
Template ID 256																	Field Count = 5																																
0	sourceIPv4Address = 8																Field Length = 4																																
0	destinationIPv4Address = 12																Field Length = 4																																
0	ipNextHopIPv4Address = 15																Field Length = 4																																
0	packetDeltaCount = 2																Field Length = 4																																
0	octetDeltaCount = 1																Field Length = 4																																

A.2.2. Template Set Using Enterprise-Specific Information Elements

We want to report the following Information Elements:

- The IPv4 source IP address: sourceIPv4Address in [IPFIX-IANA], with a length of 4 octets

- The IPv4 destination IP address: destinationIPv4Address in [IPFIX-IANA], with a length of 4 octets
- An enterprise-specific Information Element representing proprietary information, with a type of 15 and a length of 4
- The number of packets of the Flow: packetDeltaCount in [IPFIX-IANA], with a length of 4 octets
- The number of octets of the Flow: octetDeltaCount in [IPFIX-IANA], with a length of 4 octets

Therefore, the Template Set will be composed of the following:

0																1																2																3															
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9																								
Set ID = 2																Length = 32 octets																																															
Template ID 257																Field Count = 5																																															
sourceIPv4Address = 8																Field Length = 4																																															
destinationIPv4Address = 12																Field Length = 4																																															
Information Element Id. = 15																Field Length = 4																																															
Enterprise number																																																															
packetDeltaCount = 2																Field Length = 4																																															
octetDeltaCount = 1																Field Length = 4																																															

A.3. Data Set Example

In this example, we report the following three Flow Records:

Src IP addr.	Dst IP addr.	Next Hop addr.	Packet Number	Octets Number
192.0.2.12	192.0.2.254	192.0.2.1	5009	5344385
192.0.2.27	192.0.2.23	192.0.2.2	748	388934
192.0.2.56	192.0.2.65	192.0.2.3	5	6534

0

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

1

2

3

Set ID = 256

Length = 64

192.0.2.12

192.0.2.254

192.0.2.1

5009

5344385

192.0.2.27

192.0.2.23

192.0.2.2

748

388934

192.0.2.56

192.0.2.65

192.0.2.3

5

6534

Note that padding is not necessary in this example.

A.4. Options Template Set Examples

A.4.1. Options Template Set Using IANA Information Elements

Per line card (the router being composed of two line cards), we want to report the following Information Elements:

- Total number of IPFIX Messages: exportedMessageTotalCount [IPFIX-IANA], with a length of 2 octets
- Total number of exported Flows: exportedFlowRecordTotalCount [IPFIX-IANA], with a length of 2 octets

The line card, which is represented by the lineCardId Information Element [IPFIX-IANA], is used as the Scope Field.

Therefore, the Options Template Set will be:

0										1										2										3									
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1								
Set ID = 3										Length = 24																													
Template ID 258										Field Count = 3																													
Scope Field Count = 1										lineCardId = 141																													
Scope 1 Field Length = 4										exportedMessageTotalCount=41																													
Field Length = 2										exportedFlowRecordTotalCo.=42																													
Field Length = 2										Padding																													

A.4.2. Options Template Set Using Enterprise-Specific Information Elements

Per line card (the router being composed of two line cards), we want to report the following Information Elements:

- Total number of IPFIX Messages: exportedMessageTotalCount [IPFIX-IANA], with a length of 2 octets
- An enterprise-specific number of exported Flows, with a type of 42 and a length of 4 octets

The line card, which is represented by the lineCardId Information Element [IPFIX-IANA], is used as the Scope Field.

The format of the Options Template Set is as follows:

```

      0          1          2          3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Set ID = 3           |           Length = 28           |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Template ID 259       |           Field Count = 3       |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Scope Field Count = 1   | 0 |           lineCardId = 141 |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Scope 1 Field Length = 4 | 0 | exportedFlowRecordTotalCo.=41 |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Field Length = 2       | 1 | Information Element Id. = 42 |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Field Length = 4       |           Enterprise number   ...
+-----+-----+-----+-----+-----+-----+-----+-----+
...           Enterprise number   |           Padding             |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

A.4.3. Options Template Set Using an Enterprise-Specific Scope

In this example, we want to export the same information as in the example in Section A.4.1:

- Total number of IPFIX Messages: exportedMessageTotalCount [IPFIX-IANA], with a length of 2 octets
- Total number of exported Flows: exportedFlowRecordTotalCount [IPFIX-IANA], with a length of 2 octets

But this time, the information pertains to a proprietary scope, identified by enterprise-specific Information Element number 123.

The format of the Options Template Set is now as follows:

```

      0               1               2               3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Set ID = 3           |           Length = 28           |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Template ID 260       |           Field Count = 3       |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Scope Field Count = 1   | 1 | Scope 1 Infor. El. Id. = 123 |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Scope 1 Field Length = 4 |           Enterprise Number ...
+-----+-----+-----+-----+-----+-----+-----+-----+
...           Enterprise Number    | 0 | exportedMessageTotalCount=41 |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Field Length = 2       | 0 | exportedFlowRecordTotalCo.=42 |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Field Length = 2       |           Padding              |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

A.4.4. Data Set Using an Enterprise-Specific Scope

In this example, we report the following two Data Records:

Enterprise field 123	IPFIX Message	Exported Flow Records
1	345	10201
2	690	20402


```

      0               1               2               3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Set ID = 260           |           Length = 20           |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                   |           1           |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           345                     |           10201         |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                   |           2           |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           690                     |           20402         |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

A.5. Variable-Length Information Element Examples

A.5.1. Example of Variable-Length Information Element with Length Inferior to 255 Octets

```

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

```

A.5.2. Example of Variable-Length Information Element with 3 Octet Length Encoding

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|           255           |           1000           |           IE ...           |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           1000 octet Information Element           |
+-----+-----+-----+-----+-----+-----+-----+-----+
:                                     ...                                     :
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     ... IE           |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

References

Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3436] Jungmaier, A., Rescorla, E., and M. Tuexen, "Transport Layer Security over Stream Control Transmission Protocol", RFC 3436, December 2002.
- [RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 3629, November 2003.
- [RFC3758] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", RFC 3758, May 2004.
- [RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, September 2007.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.
- [RFC5905] Mills, D., Delaware, U., Martin, J., Burbank, J. and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010
- [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, March 2011.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.
- [RFC6520] Seggelmann, R., Tuexen, M., and Williams, M., "Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension", RFC 6520, February 2012.

- [TCP] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.
- [UDP] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.
- [RFC5102bis] Quittek, J., Bryant S., Claise, B., Aitken, P., and J. Meyer, "Information Model for IP Flow Information Export", draft-claise-ipfix-information-model-rfc5102bis-01.txt, Work in Progress, October 2011.
- [IPFIX-IANA] <http://www.iana.org/assignments/ipfix/ipfix.xml>

Informative References

- [RFC2579] McCloghrie, K., Perkins, D., and J. Schoenwaelder, "Textual Conventions for SMIV2", STD 58, RFC 2579, April 1999.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC3917] Quittek, J., Zseby, T., Claise, B., and S. Zander, "Requirements for IP Flow Information Export (IPFIX)", RFC 3917, October 2004.
- [RFC3954] Claise, B., Ed., "Cisco Systems NetFlow Services Export Version 9", RFC 3954, October 2004.
- [RFC5101] Claise, B., Ed., "Bidirectional Flow Export Using IP Flow Information Export (IPFIX)", RFC 5103, January 2008.
- [RFC5103] Trammell, B., and E. Boschi, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information", RFC 5101, January 2008.
- [RFC5153] Boschi, E., Mark, L., Quittek J., and P. Aitken, "IP Flow Information Export (IPFIX) Implementation Guidelines", RFC5153, April 2008
- [RFC5470] Sadasivan, G., Brownlee, N., Claise, B., and J. Quittek, "Architecture for IP Flow Information Export", RFC5470, March 2009.

- [RFC5472] Zseby, T., Boschi, E., Brownlee, N., and B. Claise, "IP Flow Information Export (IPFIX) Applicability", RFC5472, March 2009.
- [RFC5471] Schmoll, C., Aitken, P., and B. Claise, "Guidelines for IP Flow Information Export (IPFIX) Testing", RFC5471, March 2009.
- [RFC5473] Boschi, E., Mark, L., and B. Claise, "Reducing Redundancy in IP Flow Information Export (IPFIX) and Packet Sampling (PSAMP) Reports", RFC5473, March 2009.
- [RFC5474] Duffield, N., Ed., 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", RFC5476, March 2009.
- [RFC5477] Dietz, T., Claise, B., Aitken, P., Dressler, F., and G. Carle, "Information Model for Packet Sampling Exports", RFC 5477, March 2009.
- [RFC5610] Boschi, E., Trammell, B., Mark, L., and T. Zseby, "Exporting Type Information for IP Flow Information Export (IPFIX) Information Elements", RFC 5610, July 2009.
- [RFC5655] Trammell, B., Boschi, E., Mark, L., Zseby, T., and A. Wagner, "Specification of the IP Flow Information Export (IPFIX) File Format", RFC 5655, October 2009.
- [RFC6083] Tuexen, M., Seggelman, R. and E. Rescola, "Datagram Transport Layer Security (DTLS) for Stream Control Transmission Protocol (SCTP)", RFC6083, January 2011.
- [RFC6313] Claise, B., Dhandapani, G., Aitken, P, and S. Yates, "Export of Structured Data in IP Flow Information Export (IPFIX)", RFC6313, July 2011.
- [RFC6183] Kobayashi, A., Claise, B., Muenz, G, and K. Ishibashi, "IP Flow Information Export (IPFIX) Mediation: Framework", RFC6183, April 2011.
- [RFC6526] Claise, B., Aitken, P., Johnson, A. and G. Muenz, "IPFIX Export per SCTP Stream", RFC 6526, March 2012.

- [RFC6528] Gont, F. and S. Bellovin, "Defending Against Sequence Number Attacks", RFC 6528, February 2012.
- [RFC6615] Dietz, T., Kobayashi, A., Claise, B., and G. Muenz, "Definitions of Managed Objects for IP Flow Information Export", RFC 6615, June 2012.
- [RFC6727] Dietz, T. Ed., Claise, B., and J. Quittek, "Definitions of Managed Objects for Packet Sampling", RFC 6727, October 2012.
- [RFC6728] Muenz, G., Claise, B., and P. Aitken, "Configuration Data Model for IPFIX and PSAMP", RFC 6728, October 2012.
- [PEN-IANA] IANA Private Enterprise Numbers registry
<http://www.iana.org/assignments/enterprise-numbers>.
- [POSIX.1] IEEE 1003.1-2008 - IEEE Standard for Information Technology - Portable Operating System Interface, IEEE, 2008.
- [IEEE.754.1985] Institute of Electrical and Electronics Engineers, "Standard for Binary Floating-Point Arithmetic", IEEE Standard 754, August 1985.
- [UTF8-EXPLOIT] Davis, M. and M. Suignard, "Unicode Technical Report #36: Unicode Security Considerations", The Unicode Consortium, July 2012.
- [IPFIX-MED-PROTO] Claise, B., Kobayashi, A., and B. Trammell, "Specification of the Protocol for IPFIX Mediations", draft-ietf-ipfix-mediation-protocol-03, Work in Progress, January 2013.

Acknowledgments

We would like to thank Ganesh Sadasivan, as well, for his significant contribution during the initial phases of the protocol specification. Additional thanks to Juergen Quittek for the coordination job within IPFIX and PSAMP; Nevil Brownlee, Dave Plonka, and Andrew Johnson for the thorough reviews; Randall Stewart and Peter Lei for their SCTP expertise and contributions; Martin Djernaes for the first essay on the SCTP section; Michael Behringer and Eric Vyncke for their advice and knowledge in security; Michael Tuexen for his help regarding the DTLS section; Elisa Boschi for her contribution regarding the improvement of SCTP sections; Mark Fullmer, Sebastian Zander, Jeff Meyer, Maurizio Molina, Carter Bullard, Tal Givoly, Lutz Mark, David Moore, Robert Lowe, Paul Calato, Andrew Feren, Gerhard Muenz, and many more, for the technical reviews and feedback.

Authors' Addresses

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

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

Brian Trammell (Ed.)
Swiss Federal Institute of Technology Zurich
Gloriastrasse 35
8092 Zurich
Switzerland

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

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

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

Contributors' Addresses

Stewart Bryant
Cisco Systems, Inc.
250, Longwater,
Green Park,
Reading, RG2 6GB,
United Kingdom

Phone: +44 (0)20 8824-8828
EMail: stbryant@cisco.com

Simon Leinen
SWITCH
Werdstrasse 2
P.O. Box
8021 Zurich
Switzerland

Phone: +41 44 268 1536
EMail: simon.leinen@switch.ch

Thomas Dietz
NEC Europe Ltd.
NEC Laboratories Europe
Network Research Division
Kurfuersten-Anlage 36
69115 Heidelberg
Germany

Phone: +49 6221 4342-128
EMail: Thomas.Dietz@nw.necclab.eu

IPFIX Working Group
Internet-Draft
Intended Status: Standards Track
Expires May 13, 2013

C. Inacio
Carnegie Mellon University
November 9, 2012

Private Enterprise Information Elements Registry Exchange
<draft-inacio-ipfix-penie-00.txt>

Abstract

This extension to the IPFIX protocol is intended to provide a mechanism for IPFIX exporters which export private information elements to also transmit information to the collectors. The mechanism is designed to be able to send a URI with information about the private information elements via an options template.

Status of this Memo

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

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

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

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

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

This Internet-Draft will expire in May 2013.

Copyright Notice

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

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

carefully, as they describe your rights and restrictions with respect to this document.

Table of Contents

1. Introduction	4
2. Options Record Format	4
3. Registry Design	5
3.1. Registry Introduction	6
3.2. Registry Informational Elements	6
3.3. Registry Formatting	7
6. IANA Considerations	7
6. References	7
6.1. Normative References	7
6.2. Informative References	8
Authors' Addresses	8
Appendix A.	8
99.0. To be removed	10
99.1. Formatting End of Page	12


```

+-----+
|      Field Length = 65536      |
+-----+

```

Figure 1: Example PENIE Template Definition

[illegible]

Figure 2: Example PENIE Data Record

The options record allows for creating a URI reference for a private enterprise number. It is important to note that more than one URI per a single private enterprise number. The burden of resolving all declared registries falls onto the collector to be able to decode all information elements received from the exporters.

3. Registry Design

3.1. Registry Introduction

The registry design goals are to capture all the information that IPFIX Type Information [RFC5610] provides about individual elements, but to also present more metadata about both individual elements as well as have the ability to provide more information about a collection of elements.

3.2. Registry Informational Elements

The top level of a registry contains the following additional elements:

- o Registry ID - This is an ID that can be used by the creator of the registry to be able to track the registry as a unique item.
- o Version - Indicates the release version of the registry.
- o Name - A common name that can be used to refer to the registry.
- o Security Type - This is a new entry type that allows the complete set of elements defined in the registry to be contained within a security type class. By allowing the collector to understand the security type, if present, of the information elements a new class of actions may be taken by a collector implementation.
- o Policy Type - Similar to the security type, this new entry allows the complete set of elements defined in the registry to be contained within a policy type class. Again, similar to the security type class, the collector may take new actions based upon understanding the policy type of an information element.
- o Canonical URI - This is the canonical URI to be able to locate the authoritative version of the registry.
- o Root EID - This defines the Private Enterprise ID for all elements defined in the registry.
- o Copyright - Optionally the copyright information for the registry
- o Contact - Contact information to be able to contact the publisher of the registry.
- o Directory - (I can't remember, but its a URI).

Each information element defined in the registry are as follows:

- o ID - The information element ID.
- o PEN - The private enterprise number.
- o Data type - The data type of the element, as defined in RFC 5610.
- o Semantics - The semantic of the element, as defined in RFC 5610.
- o Units - The units of the element, as defined in RFC 5610.
- o Description - The human readable (and hopefully understandable) description of the element.
- o MIME Path - An optional MIME path definition of the element.

3.3. Registry Formatting

XML or JSON or ??? format to be added here.

5. Security Considerations

There are no security considerations relevant to this document, beyond the security considerations necessary in the IPFIX protocol specification [RFC5101] and its successors.

6. IANA Considerations

IANA needs to create two registries with expert review:

Security types Policy types

and create a code point for a new information element.

6. References

6.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC5101] Claise, B., "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information", RFC 5101, January 2008.
- [RFC5102] Quittek, J., Bryant, S., Claise, B., Aitken, P., and Meyer, J., "Information Model for IP Flow Information Export",

RFC 5102, January 2008.

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

[RFC4234] Crocker, D. and P. Overell, "Augmented BNF for Syntax
Specifications: ABNF", RFC 4234, October 2005.

6.2. Informative References

[2223BIS] Reynolds, J. and R. Braden, "Instructions to Request for
Comments (RFC) Authors", draft-rfc-editor-rfc2223bis-
08.txt, August 2004.

Authors' Addresses

Christopher Inacio
Carnegie Mellon University
4500 5th Avenue
Pittsburgh, PA 15213
USA

EMail: inacio@sei.cmu.edu

Appendix A.

Relax-NG based definition of a proposed schema. Can be processed with trang
to create a well defined XML XSD file.

```
namespace r = "http://www.ietf.org/ipfix/ipfix-private-element-registry/1.0"
```

```
# It's unclear what the right way of referencing an info element ought  
# to be. By PEN/ID pair? By some XML ID? Both has problems. Leave it  
# text for now.  
info-elem-ref = text
```

```
r-data-type = element r:data-type {  
  "octetArray" |  
  "unsigned8" |  
  "unsigned16" |  
  "unsigned32" |  
  "unsigned64" |  
  "signed8" |  
  "signed16" |  
  "signed32" |  
  "signed64" |  
  "float32" |  
  "float64" |
```

```
"boolean" |
"macAddress" |
"string" |
"dateTimeSeconds" |
"dateTimeMilliseconds" |
"dateTimeMicroseconds" |
"dateTimeNanoseconds" |
"ipv4Address" |
"ipv6Address" |
"basicList" |
"subTemplateList" |
"subTemplateMultiList"
}

r-semantics = element r:semantics {
  "default" |
  "quantity" |
  "totalCounter" |
  "deltaCounter" |
  "identifier" |
  "flags" |
  "list"
}

r-units = element r:units {
  "none" |
  "bits" |
  "octets" |
  "packets" |
  "flows" |
  "seconds" |
  "milliseconds" |
  "microseconds" |
  "nanoseconds" |
  "4-octet words" |
  "messages" |
  "hops" |
  "entries"
}

#r-value-map = element r:value-map {
#   attribute type {"integer" | "text" },
#   {element value }
#}

r-element = element r:element {
  element r:id { xsd:integer {minInclusive="0" maxInclusive="65535"}} &
  element r:private-enterprise-number { xsd:integer {minInclusive="0" maxInclu
sive="4294967295"}}? &
```

```

    r-data-type &
    r-semantics? &
    r-units? &
    element r:range-begin { text }? &
    element r:range-end { text }? &
    element r:name { xsd:token } &
    element r:description { text } &

    element r:mime-path { text }?

# element r:value-map {
#     attribute type {"integer" | "text"},
# }
}

r-registry = element r:registry {
    element r:id { xsd:anyURI } &
    element r:name { text } &

    # Should these two be required or optional?
    element r:security-type { text } &
    element r:policy-type { text } &

    element r:url { xsd:anyURI } &
    element r:root-eid { xsd:integer } &
    (r-registry | r-element)*
}

r-enterprise-registry = element r:enterprise-registry {
    element r:name { text } &
    element r:copyright { text } &
    element r:contact { text } &
    element r:private-enterprise-number { xsd:integer } &
    element r:directory { xsd:anyURI } &
    r-registry*
}

start = r-enterprise-registry

```

99.0. To be removed

The RFC Editor generally uses the simplest nroff features, basically the "-ms" macro package and the following few basic nroff directives:

DIRECTIVE	FUNCTION
.ce	Center following line.

.ti # 'temporary indent' -- # is number of spaces.
 Indents only the line immediately following.

.in # Change indentation to # spaces

.nf 'No fill': begin block of text to be displayed.

.fi Fill (i.e., left-justify, line wrap)

.ne # 'need' -- Keep following # lines on same page

.bp Break page

.br Break line

.KS 'Keep Start' -- lines up to .KE on same page

.KE 'Keep End' -- end of 'keep' block

Nroff also has a '.sp' (space) directive to insert a blank line. However, it is far easier (and more readable) to use the fact that each blank line in the nroff source creates a blank line in the output.

Nroff includes many variations on the trivial commands shown above. For example, indentation can be specified relative to the current indentation, using '.in +#' or '.in -#'. Authors are welcome to use such features, but for simplicity this template uses only the simplest set of commands.

Some authors who are proficient in nroff will wish to use more advanced features, including perhaps their own macros. This is a private matter for the author, unless and until the document is submitted to the RFC Editor for publication as an RFC. Upon document submission, the RFC Editor will request the nroff source, if any. If the source is sufficiently straightforward, it will be used by the RFC Editor to speed the publication process. If not, the RFC Editor will generate a new nroff source, generally using the simple subset above.

The considerations here are as follows:

- o Defined macros (beyond the -ms package) must be in-line at the front of the source. The RFC Editor is currently prepared to maintain only one source file for each published RFC.
- o Some of the editors are not nroff experts, and even those who may be do not have the time to figure out some complex/obscure

macro. If any special knowledge about these macros is needed to modify the text for editorial purposes, the RFC Editor will find it more expedient to generate a new .nroff source for the document.

- o The RFC Editor does not keep a distinct Make file for each RFC, so it is not helpful to send us a tar file or shar script that magically makes a directory and builds an RFC. Our primary input is a .txt file, with a .nroff file as a possible secondary input. When the RFC is published, the RFC Editor will archive a .txt file and a corresponding \&.nroff file.

In other words, keep it simple and you can help us a lot; don't show off your programming prowess and waste our time.

99.1. Formatting End of Page

The Unix command to create a formatted Internet Draft is:

```
"nroff -ms input-file.nroff > output-file.txt"
```

However, nroff will not follow the RFC standard format for a page: a Form feed (FF or Control-L)) after the last visible line on the page and no extra line feeds before the first visible line of the next page. We want:

```
last visible line on page i
^L
first visible line on page i+1
```

We invented hacks to fix this. The original hack was a "sed" script that called a "C" program called "pg". More recently, we have been using a simple Perl script (see Appendix A). Then the command to process the nroff source file becomes:

```
nroff -ms input-file.nroff | fix.pl > output-file.txt
```

For example:

```
nroff -ms 2-nroff.template | fix.pl > 2-nroff.template.txt
```

IPFIX
Internet Draft
Intended status: Informational
Expires: August 2013
February 24, 2013

R. Krishnan
D. Meyer
Brocade Communications
Ning So
Tata Communications

Flow Aware Packet Sampling Techniques

draft-krishnan-ipfix-flow-aware-packet-sampling-03.txt

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, except to publish it as an RFC and to translate it into languages other than English.

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

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

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

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

This Internet-Draft will expire on August 24, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Abstract

The demands on the networking infrastructure and thus the switch/router bandwidths are growing exponentially; the drivers are bandwidth hungry rich media applications, inter data center communications etc. Using sampling techniques, for a given sampling rate, the amount of samples that need to be processed is increasing exponentially. This draft suggests flow aware sampling techniques for handling various scenarios with minimal sampling overhead.

Table of Contents

1. Introduction.....	2
1.1. Acronyms.....	3
1.2. Terminology.....	3
2. Flow Aware Packet Sampling.....	3
2.1. Large Flow Recognition.....	4
2.1.1. Flow Identification.....	4
2.1.2. Criteria for Identifying a Large Flow.....	5
2.1.3. Automatic Recognition.....	5
2.1.3.1. Applicability of suggested technique.....	6
2.1.3.2. Enhancements to suggested technique.....	7
2.1.3.3. Handling Inactive Large Flows.....	7
2.1.4. Simulation.....	7
3. Acknowledgements.....	7
4. IANA Considerations.....	7
5. Security Considerations.....	7
6. Data Model Considerations.....	8
7. References.....	8
7.1. Normative References.....	8
7.2. Informative References.....	8

1. Introduction

Packet sampling techniques in switches and routers provide an effective mechanism for approximate detection of various types of flows -- long-lived large flows and other flows (which include long-lived small flows, short-lived small/large flows) with minimal packet replication bandwidth overhead. A large percentage of the packet samples comprise of long-lived large flows and a small percentage of the packet samples comprise of other flows. The long-lived large flows aka top-talkers consume a large percentage of the bandwidth and small percentage of the flow space. The other flows, which are the typical cause of security threats like Denial of Service (DOS) attacks, Scanning attacks etc., consume a small percentage of the bandwidth and a large percentage of the flow space. This draft

explores light-weight techniques for automatically detecting the top-talkers in real-time with a high degree of accuracy and sampling only the other flows -- this makes security threat detection more effective with minimal sampling overhead.

1.1. Acronyms

DOS: Denial of Service

GRE: Generic Routing Encapsulation

MPLS: Multi Protocol Label Switching

NVGRE: Network Virtualization using Generic Routing Encapsulation

TCAM: Ternary Content Addressable Memory

STT: Stateless Transport Tunneling

VXLAN: Virtual Extensible LAN

1.2. Terminology

Large flow(s): long-lived large flow(s)

Small flow(s): long-lived small flow(s) and short-lived small/large flow(s)

2. Flow Aware Packet Sampling

The steps in flow aware packet sampling are described below

1) Large Flow Recognition in switches and routers:

From a bandwidth and time duration perspective, in order to identify large flows in switches and routers, we define an observation interval and observe the bandwidth of the flow over that interval. A flow that exceeds a certain minimum bandwidth threshold over that observation interval would be considered a large flow. For identifying large flows, use the techniques described in Section 2.1. This helps in identifying the large flows aka top-talkers in real-time with a high degree of accuracy in switches and routers.

2) Large Flow Classification:

The identified large flows can be broadly classified into 2 categories as detailed below.

- a. Well behaved (steady rate) large flows, e.g. video streams
- b. Bursty (fluctuating rate) large flows e.g. Peer-to-Peer traffic

The large flows can be sampled at a low rate for further analysis or need not be sampled. If desired, the large flows could be exported to a central entity, for e.g. Netflow Collector, for further analysis.

3) Small Flow Processing:

The small flows (excluding the large flows) can be sampled at a normal rate. The small flows can be examined for determining security threats like DOS attacks (for e.g. SYN floods), Scanning attacks etc. [FDDOS, PDSN, ALDS]

Thus, we can see that, security threat detection is possible with minimal sampling overhead.

For packet sampling, it is recommended to use PSAMP -- [RFC 5474], [RFC 5475], [RFC 5476], [RFC 5477].

2.1. Large Flow Recognition

2.1.1. Flow Identification

A flow (large flow or small flow) can be defined as a sequence of packets for which ordered delivery should be maintained. Flows are typically identified using one or more fields from the packet header from the following list:

- . Layer 2: source MAC address, destination MAC address, VLAN ID.
- . IP header: IP Protocol, IP source address, IP destination address, flow label (IPv6 only), TCP/UDP source port, TCP/UDP destination port.
- . MPLS Labels.

For tunneling protocols like GRE, VXLAN, NVGRE, STT, etc., flow identification is possible based on inner and/or outer headers. The above list is not exhaustive. The mechanisms described in this

document are agnostic to the fields that are used for flow identification.

2.1.2. Criteria for Identifying a Large Flow

From a bandwidth and time duration perspective, in order to identify large flows we define an observation interval and observe the bandwidth of the flow over that interval. A flow that exceeds a certain minimum bandwidth threshold over that observation interval would be considered a large flow.

The two parameters -- the observation interval, and the minimum bandwidth threshold over that observation interval -- should be programmable in a switch or a router to facilitate handling of different use cases and traffic characteristics. For example, a flow which is at or above 10 Mbps for a time period of at least 30 minutes could be declared a large flow.

An optional parameter is a policy specification (for e.g. identify flows only from a given IP source and/or destination address)

2.1.3. Automatic Recognition

Implementations can perform automatic recognition of large flows in a switch or a router -- it is an inline solution and would be expected to operate at line rate.

The advantages and disadvantages of automatic recognition are:

Advantages:

- . Accurate and performed in real-time.

Disadvantages:

- . Not supported in many switches and routers.

As mentioned earlier, the observation interval for determining a large flow and the bandwidth threshold for classifying a flow as a large flow should be programmable parameters in a switch or a router.

The implementation of automatic recognition of large flows is vendor dependent. Below is a suggested technique.

This technique uses a counting Bloom filter using thresholding and periodic reset. This technique requires a few tables -- a flow table, and multiple hash tables.

The flow table comprises entries which are programmed with packet fields for flows that are already known to be large flows and each entry has a corresponding byte counter. It is initialized as an empty table (i.e. none of the incoming packets would match a flow table entry).

The hash tables each have a different hash function and comprise entries which are byte counters. The counters are initialized to zero and would be modified as described by the algorithm below.

Step 1) If the large flow exists in the flow table (for e.g. TCAM), increment the counter associated with the flow by the packet size. Else, proceed to Step 2.

Step 2) The hash function for each table is applied to the fields of the packet header and the result is looked up in parallel in corresponding hash table and the associated counter corresponding to the entry that is hit in that table is incremented by the packet size. If the counter exceeds a programmed byte threshold in the observation interval (this counter threshold would be set to match the bandwidth threshold) in the entries that were hit in all of the hash tables, a candidate large flow is learnt and programmed in the flow table and the counters are reset.

Additionally, the counters in all of the hash tables must be reset every observation interval.

There may be some false positives due to multiple small flows masquerading as a large flow. The number of such false positives is reduced by increasing the number of parallel hash tables using different hash functions. There will be a design tradeoff between size of the hash tables, the number of hash tables, and the probability of a false positive.

This technique for automatic recognition is also suggested in [draft-krishnan-opsawg-large-flow-load-balancing] -- please refer to the draft for more details on the algorithm.

2.1.3.1. Applicability of suggested technique

The suggested technique for automatic recognition works well for standard applications generating large flows, for e.g. video content like movies and catch-up episodes, backup transactions etc. with a detection time of approximately 30-60 seconds. These detection times ensure that short-lived large flows, for e.g. HD video clips, are not unnecessarily recognized.

2.1.3.2. Enhancements to suggested technique

If faster flow recognition times are desired (much shorter than 30s), the suggested technique may pose the following problem that the effective filtered flow size is phase-dependent: that is, relatively smaller constant-rate flows, for e.g. HD video clips, beginning early within a counting Bloom filter reset interval would be unnecessarily detected with the same probability as relatively larger flows beginning toward the interval.

[VRM] suggests techniques for addressing the above problem using rotating conservative counting Bloom filters with periodic decay.

2.1.3.3. Handling Inactive Large Flows

Once a flow has been recognized as a large flow, it should continue to be recognized as a large flow as long as the traffic received during an observation interval exceeds some fraction of the bandwidth threshold, for example 80% of the bandwidth threshold. If the traffic received during an observation interval falls below a fraction of the bandwidth threshold, the large flow should be removed from the flow-table.

2.1.4. Simulation

Simulation results for flow aware packet sampling are presented in Appendix A. The goal of the simulation is to demonstrate the effectiveness of flow aware packet sampling in a multi-tenant video streaming data center.

3. Acknowledgements

The authors would like to thank Juergen Quittek, Brian Carpenter, Michael Fargano, Michael Bugenhagen, Jianrong Wong and Brian Trammell for all the support and valuable input.

4. IANA Considerations

This memo includes no request to IANA.

5. Security Considerations

This document does not directly impact the security of the Internet infrastructure or its applications. In fact, it proposes techniques which could help in identifying a DOS attack pattern.

6. Data Model Considerations

In Section 2, for exporting the identified large flows to an external entity, it is recommended to use IPFIX protocol [RFC 5101].

Section 2.1.2 defines programmable parameters in switches and routers for automatic identification. IETF could potentially consider a standards-based activity around defining a data model for moving this information from a central management entity to the switch/router.

7. References

7.1. Normative References

7.2. Informative References

[RFC 5474] N. Duffield et al., "A Framework for Packet Selection and Reporting", March 2009.

[RFC 5475] T. Zseby et al., "Sampling and Filtering Techniques for IP Packet Selection", March 2009.

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

[RFC 5477] T. Dietz et al., "Information Model for Packet Sampling Exports", March 2009.

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

[draft-krishnan-opsawg-large-flow-load-balancing] R. Krishnan et al., "Best Practices for Optimal LAG/ECMP Component Link Utilization in Provider Backbone Networks", February 2013

[VRM] G. Bianchi et al., "Measurement Data Reduction through Variation Rate Metering", INFOCOM 2010

[PDSN] Ignasi Paredes-Oliva et al., "Portscan Detection with Sampled NetFlow", TMA 2009

[ALDS] Z. Morley Mao et al., "Analyzing Large DDoS Attacks Using Multiple Data Sources", SIGCOMM 2006

[FDDOS] David Holmes, "The DDoS Threat Spectrum", F5 White paper 2012

Appendix A: Simulation of Flow aware packet sampling

Goal:

Demonstrate the effectiveness of flow aware packet sampling in a practical use case, for e.g. multi-tenant video streaming in a data center.

Test Topology:

Multiple virtual servers (server hosted on a virtual machine) connected to a virtual switch (vSwitch) which in turn connects to the data center network using a 10Gbps ethernet interface.

2 virtual servers are active.

First virtual server

. Traffic types

- o HD MPEG-4 video streams (bit rate 10Mbps) - 100 - 1Gbps
- o SD MPEG-2 video streams (bit rate 4Mbps) - 300 - 1.2Gbps
- o Other traffic - 500Mbps (Video clips, DOS attacks (for e.g. SYN floods), Scanning attacks etc.)

. Aggregate traffic - 2.7Gbps

Second virtual server

. Traffic types

- o HD MPEG-4 video streams (bit rate 10Mbps) - 50 - .5Gbps
- o SD MPEG-2 video streams (bit rate 4Mbps) - 500 - 2.0Gbps
- o Backup transaction - 100Mbps
- o Other traffic - 500Mbps (Video clips, DOS attacks (for e.g. SYN floods), Scanning attacks etc.)

. Aggregate traffic - 3.1Gbps

Total traffic on 2 servers - 5.8Gbps

Existing techniques:

Normal sampling rate - 1:1000

Total sampled traffic = $5.8\text{Gbps}/1000 = 5.8\text{Mbps}$

Flow aware sampling technique:

Large flow recognition parameters

- . Observation interval for large flow - 60 seconds
- . Minimum bandwidth threshold over the observation interval - 2Mbps

Aggregate bit rate of large flows = 4.8Gbps

Aggregate bit rate of small flows = 1Gbps

Low sampling rate of large flows - 1:10000

Normal sampling rate of small flows - 1:1000

Total sampled traffic = $4.8\text{Gbps}/10000 + 1\text{Gbps}/1000 = 1.48\text{Mbps}$

Percentage improvement in sampling (most of the samples are only small flows) = $(5.8 - 1.48)/5.8 \approx 78\%$

The small flows can be examined in a central entity like Netflow Collector for determining security threats like DOS attacks, Scanning attacks etc. Thus, we can see that, security threat detection is possible with minimal sampling overhead.

Authors' Addresses

Ram Krishnan
Brocade Communications
San Jose, 95134, USA

Phone: +001-408-406-7890
Email: ramk@brocade.com

David Meyer
Brocade Communications

San Jose, 95134, USA

Phone: +001-408-333-4193

Email: dmm@l-4-5.net

Ning So

Tata Communications

Plano, TX 75082, USA

Phone: +001-972-955-0914

Email: ning.so@tatacommunications.com

IPFIX Working Group
Internet-Draft
Intended status: Informational
Expires: May 9, 2013

B. Trammell
ETH Zurich
November 5, 2012

Textual Representation of IPFIX Abstract Data Types
draft-trammell-ipfix-text-adt-00.txt

Abstract

This document defines UTF-8 representations for IPFIX abstract data types, to support interoperable usage of the IPFIX Information Elements with protocols based on textual encodings.

Status of this Memo

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

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

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

This Internet-Draft will expire on May 9, 2013.

Copyright Notice

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

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

Table of Contents

1. Introduction	3
2. Terminology	3
3. Identifying Information Elements	3
4. Data Type Encodings	4
4.1. octetArray	4
4.2. unsigned*	4
4.3. signed*	5
4.4. float*	6
4.5. boolean	6
4.6. macAddress	6
4.7. string	6
4.8. dateTime*	6
4.9. ipv4Address	7
4.10. ipv6Address	7
4.11. basicList, subTemplateList, and subTemplateMultiList	7
5. Security Considerations	7
6. IANA Considerations	7
7. References	7
7.1. Normative References	7
7.2. Informative References	8
Appendix A. Example	8
Author's Address	10

1. Introduction

The IPFIX Information Model, as defined by the IANA IPFIX Information Element Registry, provides a rich set of Information Elements for description of information about network entities and network traffic data, and abstract data types for these Information Elements. The IPFIX Protocol Specification [I-D.ietf-ipfix-protocol-rfc5101bis], in turn, defines a big-endian binary encoding for these abstract data types suitable for use with the IPFIX Protocol.

However, present and future operations and management protocols and applications may use textual encodings, and generic framing and structure as in JSON or XML. A definition of canonical textual encodings for the IPFIX abstract data types would allow this set of Information Elements to be used for such applications, and for these applications to interoperate with IPFIX applications at the Information Element definition level.

Note that templating or other mechanisms for data description for such applications and protocols are application specific, and therefore out of scope for this document: only Information Element identification and data value representation are defined here.

2. Terminology

Capitalized terms defined in the IPFIX Protocol Specification [I-D.ietf-ipfix-protocol-rfc5101bis] and the IPFIX Information Model [I-D.ietf-ipfix-information-model-rfc5102bis] are used in this document as defined in those documents. In addition, this document defines the following terminology for its own use:

Enclosing Context

Textual representation of IPFIX data values is applied to use the IPFIX Information Model within some existing textual format (e.g. XML, JSON). This outer format is referred to as the Enclosing Context within this document. Enclosing Contexts define escaping and quoting rules for represented data values.

3. Identifying Information Elements

The IPFIX Information Element Registry [iana-ipfix-assignments] defines a set of Information Elements and numbered by Information Element Identifiers, and named for human-readability. These Information Element Identifiers are meant for use with the IPFIX protocol, and have little meaning when applying the IPFIX Information Element Registry to textual representations.

Instead, applications using textual representations of Information Elements SHOULD use Information Element names to identify them; see Appendix A for examples illustrating this principle.

4. Data Type Encodings

[FIXME frontmatter]

This section uses ABNF [RFC5234], including the Core Rules in Appendix B, to describe the format of textual representations of IPFIX abstract data types.

4.1. octetArray

[FIXME: native hex strings for comparative human readability.]

4.2. unsigned*

First, in the special case that the unsigned Information Element has identifier semantics, and refers to a set of codepoints, either in an external registry, a sub-registry, or directly in the description of the Information Element, then the name or short description for that codepoint MAY be used to improve readability.

If the Enclosing Context defines a representation for unsigned integers, that representation SHOULD be used.

Otherwise, the values of Information Elements of an unsigned integer type may be represented either as unprefixd base-10 (decimal) strings, or as base-16 (hexadecimal) strings prefixed by '0x'; in ABNF:

```
unsigned = 1*DIGIT / '0x' 1*HEXDIG
```

Leading zeroes are allowed in either encoding, and do not signify base-8 (octal) encoding.

The encoded value must be in range for the corresponding abstract data type or Information Element. Out of range values should be interpreted as clipped to the implicit range for the Information Element as defined by the abstract data type, or to the explicit range of the Information Element if defined. Minimum and maximum values for abstract data types are shown in Table 1 below.

type	minimum	maximum
unsigned8	0	255
unsigned16	0	65536
unsigned32	0	4294967295
unsigned64	0	18446744073709551615

Table 1: Ranges for unsigned abstract data types

4.3. signed*

If the Enclosing Context defines a representation for signed integers, that representation should be used.

Otherwise, the values of Information Elements of signed integer types should be represented as optionally-prefixed base-10 (decimal) strings; if the sign is omitted, it is assumed to be positive. In ABNF:

```
sign = "+" / "-"
```

```
signed = [sign] 1*DIGIT
```

Leading zeroes are allowed, and do not signify base-8 (octal) encoding.

The encoded value must be in range for the corresponding abstract data type or Information Element. Out of range values should be interpreted as clipped to the implicit range for the Information Element as defined by the abstract data type, or to the explicit range of the Information Element if defined. Minimum and maximum values for abstract data types are shown in Table 2 below.

type	minimum	maximum
signed8	-128	+127
signed16	-32768	+32767
signed32	-2147483648	+2147483647
signed64	-9223372036854775808	+9223372036854775807

Table 2: Ranges for signed abstract data types

4.4. float*

If the Enclosing Context defines a representation for floating point numbers, that representation should be used.

[FIXME: there appears to be no defined (non-interchange) format for floating point numbers, but we probably want to define something reasonably human-readable without getting too into locale issues.]

exponent = 'e' 1*3DIGIT

right-decimal = '.' 0*DIGIT

float = [sign] 1*DIGIT [right-decimal] [exponent]

4.5. boolean

[FIXME: frontmatter. note that booleans may also be naturally represented by the presence or absence of a value in the structure of the document in the Enclosing Context.]

boolean-yes = "l" / "y" / "Y" / "t" / "T"

boolean-no = "0" / "n" / "N" / "f" / "F"

boolean = boolean-yes / boolean-no

4.6. macAddress

[FIXME: frontmatter]

macaddress = 2*HEXDIG 5*(":" 2*HEXDIG)

4.7. string

As Information Elements of the string type are simply UTF-8 encoded strings, they are represented directly, subject to the escaping and encoding rules of the Enclosing Context. If the Enclosing Context cannot natively represent UTF-8 characters, the escaping facility provided by the Enclosing Context must be used for non-representable characters. Additionally, strings containing characters reserved in the Enclosing Context (e.g. markup characters, quotes) must be escaped or quoted according to the rules of the Enclosing Context.

4.8. dateTime*

[FIXME: [RFC3339]]

[FIXME: explain precision rules]

4.9. ipv4Address

[FIXME: frontmatter. dotted-quad.]

ipv4address = 1*3DIGIT 3*("." 1*3DIGIT)

4.10. ipv6Address

[FIXME: section 2.2 of [RFC4291], recommend section 4 of [RFC5952]]

4.11. basicList, subTemplateList, and subTemplateMultiList

These abstract data types, defined for IPFIX Structured Data [RFC6313], do not represent actual data types; they are instead designed to provide a mechanism by which complex structure below the template level. It is assumed that protocols using textual Information Element representation will provide their own structure. Therefore, Information Elements of these Data Types should not be used in textual representations.

5. Security Considerations

[FIXME: content would be nice]

6. IANA Considerations

This document has no considerations for IANA.

7. References

7.1. Normative References

- [I-D.ietf-ipfix-protocol-rfc5101bis]
Claise, B. and B. Trammell, "Specification of the IP Flow Information eXport (IPFIX) Protocol for the Exchange of Flow Information", draft-ietf-ipfix-protocol-rfc5101bis-02 (work in progress), June 2012.
- [RFC3339] Klyne, G., Ed. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, July 2002.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.

- [RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.
- [RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, August 2010.
- [iana-ipfix-assignments]
Internet Assigned Numbers Authority, "IP Flow Information Export Information Elements
(<http://www.iana.org/assignments/ipfix/ipfix.xml>)",
November 2012.

7.2. Informative References

- [I-D.ietf-ipfix-information-model-rfc5102bis]
Claise, B. and B. Trammell, "Information Model for IP Flow Information eXport (IPFIX)",
draft-ietf-ipfix-information-model-rfc5102bis-06 (work in progress), October 2012.
- [I-D.ietf-ipfix-ie-doctors]
Trammell, B. and B. Claise, "Guidelines for Authors and Reviewers of IPFIX Information Elements",
draft-ietf-ipfix-ie-doctors-07 (work in progress),
October 2012.
- [RFC6313] Claise, B., Dhandapani, G., Aitken, P., and S. Yates,
"Export of Structured Data in IP Flow Information Export (IPFIX)", RFC 6313, July 2011.

Appendix A. Example

In this section, we examine an IPFIX Template and a Data Record defined by that Template, and show how that Data Record would be represented in JSON according to the specification in this document. Note that this is specifically NOT a recommendation for a particular representation, merely an illustration of the encodings in this document.

[FIXME improve frontmatter] Figure 1 shows a Template in IESpec format as defined in section 9.1 of [I-D.ietf-ipfix-ie-doctors]. A Message containing this Template and a Data Record is shown in Figure 2, and a corresponding JSON Object using the text format defined in this document is shown in Figure 3.

```

flowStartMilliseconds(152)<dateTimeMilliseconds>[8]
flowEndMilliseconds(153)<dateTimeMilliseconds>[8]
octetDeltaCount(1)<unsigned64>[4]
packetDeltaCount(2)<unsigned64>[4]
sourceIPv6Address(27)<ipv4Address>[4]{key}
destinationIPv6Address(28)<ipv4Address>[4]{key}
sourceTransportPort(7)<unsigned16>[2]{key}
destinationTransportPort(11)<unsigned16>[2]{key}
protocolIdentifier(4)<unsigned8>[1]{key}
tcpControlBits(6)<unsigned8>[1]
flowEndReason(136)<unsigned8>[1]

```

Figure 1: Sample flow template (IPFIX)

1										2										3										4										5										6																				
0	2	4	6	8	0	2	4	6	8	0	2	4	6	8	0	2	4	6	8	0	2	4	6	8	0	2	4	6	8	0	2	4	6	8	0	2																																		
0x000a										length 135										export time 1352140263																				msg																														
sequence 0																				domain 1																				hdr																														
SetID 2										length 52										tid 256										fields 11										tmpl																														
IE 152										length 8										IE 153										length 8										set																														
IE 1										length 4										IE 2										length 4																																								
IE 27										length 16										IE 28										length 16																																								
IE 7										length 2										IE 11										length 2																																								
IE 4										length 1										IE 6										length 1																																								
IE 136										length 1										SetID 256										length 83										data																														
start time																														1352140261135										set																														
end time																														1352140262880																																								
octets										195383										packets										88																																								
sip6																																																																						
																				2001:0db8:000c:1337:0000:0000:0000:0002																																																		
dip6																																																																						
																				2001:0db8:000c:1337:0000:0000:0000:0003																																																		
sp										80										dp										32991										prt 6										tcp 19										fe 3										

Figure 2: IPFIX message containing sample flow

```
{
  "flowStartMilliseconds": "2012-11-05 18:31:01.135",
  "flowEndMilliseconds": "2012-11-05 18:31:02.880",
  "octetDeltaCount": 195383,
  "packetDeltaCount": 88,
  "sourceIPv6Address": "2001:db8:c:1337::2",
  "destinationIPv6Address": "2001:db8:c:1337::3",
  "sourceTransportPort": 80,
  "destinationTransportPort": 32991,
  "protocolIdentifier": "tcp",
  "tcpControlBits": 19,
  "flowEndReason": 3
}
```

Figure 3: JSON object containing sample flow

Author's Address

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

