IPFIX Working Group
Internet-Draft
Intended Status: Informational
Expires: April 30, 2012

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

October 30, 2011

Export of Application Information in IPFIX
draft-claise-export-application-info-in-ipfix-03

Status of this Memo

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

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

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

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-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 April 16, 2011.

Copyright Notice

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

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

Abstract

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

Conventions used in this document

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

Table of Contents

List of Figures and Tables

1. Overview

1.1. IPFIX Documents Overview

The IPFIX Protocol [RFC5101] provides network administrators with access to IP Flow information.

The architecture for the export of measured IP Flow information out of an IPFIX Exporting Process to a Collecting Process is defined in the IPFIX Architecture [RFC5470], per the requirements defined in RFC 3917 [RFC3917].

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

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

In order to gain a level of confidence in the IPFIX implementation, probe the conformity and robustness, and allow interoperability, the Guidelines for IPFIX Testing [RFC5471] presents a list of tests for implementers of compliant Exporting Processes and Collecting Processes.

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

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

2. Introduction

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

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

1. L2 protocols (such as ARP, PPP, LLDP)
2. IP protocols (such as ICMP, IGMP, GRE)
3. TCP or UDP ports (such as HTTP, Telnet, FTP)
4. Application layer header (of the application to be identified)
5. Packet data content
6. Packets and traffic behavior

The exact application identification methods are part of the Metering Process internals that aims to provide an accurate identification with a minimum false identification. This task requires a sophisticated Metering Process since the protocols do not behave in a standard manner.

1. Applications use port obfuscation where the application run on different port than the IANA assigned one. For example a HTTP server might run a TCP port 23 (assigned to telnet in [IANA-PORTS])
2. IANA does not accurately reflect how certain ports are "commonly" used today. Some ports are reserved, but

the application either never became prevalent or is not in use today.

3. The application behavior and identification logic become more and more complex

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

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

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

Application could be defined at different OSI layers, from the layer 2 to the layer 7. Examples: Cisco Discovery Protocol is layer 2 application, ICMP is layer 3 application [IANA-PROTO], HTTP is layer 4 application [IANA-PORTS], and skype is layer 7.

While an ideal solution would be an IANA registry for applications above (or inside the payload of) the well known ports [IANA-PORTS], this solution is not always possible as the some applications require well known specifications. Therefore, some reverse engineering is required, as well as a ubiquitous language for application identification. Clearly not realistic.

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

2.1. Application Information Use Cases

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

1. Network Visibility

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

2. Billing Services

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

3. Congestion Control

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

4. Security Functions

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

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

3. Terminology

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

[RFC5101], these IPFIX-specific terms have the first letter of a word capitalized when used in this document.

3.1. New Terminology

Application Tag

A unique identifier for an application.

4. applicationTag Information Element Specification

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

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

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0  1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Class. Eng. ID|         Selector ID  ...                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: applicationTag Information Element

Classification Engine ID

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

Selector ID

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

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

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

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

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

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

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

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0  1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Class. Eng. ID |            zero-valued upper-bits ...         |
```

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      ...  Selector ID                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: Selector ID encoding

4.1. Existing Classification Engine IDs

The following Engine IDs have been allocated by Cisco Systems.

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

		The Selector ID has global significance for all Cisco Systems devices under CANA governance. Hopefully the same ID will be maintained after the IANA standardization. Example: IPFIX had the port 4739 pre-assigned in the IETF draft for years. While waiting for the IANA registration, we could use this Selector ID.
	5	Reserved.
USER-Defined	6	The Selector ID represents applications defined by the user (using CLI or GUI) based on the methods described in section 2.
	7	Reserved.
	8	Reserved.
	9	Reserved.
	10	Reserved.
	11	Reserved.
CANA-L2	12	The Selector ID represents the Cisco Systems unique global layer 2 applications. The Selector ID has a global significance.
CANA-L7	13	The Selector ID represents the Cisco Systems unique global ID for the layer 7 applications. The Selector ID has global significance for all Cisco Systems devices.
	14	Reserved.
	15	Reserved.
	16	Reserved.
	17 to 254	Available.
MAX	255	255 is the maximum Engine ID.

Table 1: Existing Classification Engine IDs

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

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

4.2. Options Template Record for the Application Name

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

4.3. Resolving IANA L4 port collisions

Even if the IANA L4 ports usually point to the same protocols for both UDP, TCP or other transport types, there are some exceptions. The following table lists 10 ports that have different protocols assigned for TCP and UDP:

```
exec            512/tcp    remote process execution;
#                          authentication performed using
#                          passwords and UNIX login names
comsat/biff     512/udp    used by mail system to notify users
#                          of new mail received; currently
#                          receives messages only from
#                          processes on the same machine
login           513/tcp    remote login a la telnet;
#                          automatic authentication performed
#                          based on priviledged port numbers
#                          and distributed data bases which
#                          identify "authentication domains"
who             513/udp    maintains data bases showing who's
#                          logged in to machines on a local
#                          net and the load average of the
#                          machine
shell           514/tcp    cmd
#                          like exec, but automatic
authentication
#                          is performed as for login server
syslog          514/udp
oob-ws-https    664/tcp    DMTF out-of-band secure web services
#                          management protocol
```

```
   #                          Jim Davis
   <jim.davis&wbemsolutions.com>
   #                          June 2007
   asf-secure-rmcp 664/udp    ASF Secure Remote Management
   #                          and Control Protocol
   rfile           750/tcp
   kerberos-iv     750/udp    kerberos version iv
   submit          773/tcp
   notify          773/udp
   rpasswd         774/tcp
   acmaint_dbd     774/udp
   entomb          775/tcp
   acmaint_transd  775/udp
   busboy          998/tcp
   puparp          998/udp
   garcon          999/tcp
   applix          999/udp    Applix ac
```

Table 2: IANA layer 4 port collisions between UDP and TCP

The following table lists 19 ports that have different protocols assigned for TCP and SCTP:

```
   #               3097/tcp    Reserved
   itu-bicc-stc    3097/sctp   ITU-T Q.1902.1/Q.2150.3
                               Greg Sidebottom<gregside&home.com>
   #               5090/tcp    <not assigned>
   car             5090/sctp   Candidate AR
   #               5091/tcp    <not assigned>
   cxtp            5091/sctp   Context Transfer Protocol
                               RFC 4065 - July 2005
   #               6704/tcp    Reserved
   frc-hp          6704/sctp   ForCES HP (High Priority) channel
   #                           [RFC5811]
   #               6705/tcp    Reserved
   frc-mp          6705/sctp   ForCES MP (Medium Priority) channel
   #                           [RFC5811]
   #               6706/tcp    Reserved
   frc-lp          6706/sctp   ForCES LP (Low priority) channel
   #                           [RFC5811]
   #               9082/tcp    <not assigned>
   lcs-ap          9082/sctp   LCS Application Protocol
   #                           Kimmo Kymalainen
```

```
                                <kimmo.kymalainen&etsi.org>
                                04 June 2010
   #                9902/tcp    <not assigned>
   enrp-sctp-tls    9902/sctp   enrp/tls server channel
   #                           [RFC5353]
   #                11997/tcp   <not assigned>
   #                11998/tcp   <not assigned>
   #                11999/tcp   <not assigned>
   wmereceiving     11997/sctp  WorldMailExpress
   wmedistribution  11998/sctp  WorldMailExpress
   wmereporting     11999/sctp  WorldMailExpress
                               Greg Foutz<gregf&adminovation.com>
                                March 2006
   #                25471/tcp   <not assigned>
   rna              25471/sctp  RNSAP User Adaptation for Iurh
   #                            Dario S. Tonesi
                                <dario.tonesi&nsn.com>
                                07 February 2011
   #                29118/tcp   Reserved
   sgsap            29118/sctp  SGsAP in 3GPP
   #                29168/tcp   Reserved
   sbcap            29168/sctp  SBcAP in 3GPP
   #                29169/tcp   <not assigned>
   iuhsctpassoc     29169/sctp  HNBAP and RUA Common Association
                                John
                                Meredith<John.Meredith&etsi.org>
                                08 September 2009
   #                36412/tcp   <not assigned>
   s1-control       36412/sctp  S1-Control Plane (3GPP)
   #                            KimmoKymalainen
                                <kimmo.kymalainen&etsi.org>
                                01 September 2009
   #                36422/tcp   <not assigned>
   x2-control       36422/sctp  X2-Control Plane (3GPP)
   #                            Kimmo Kymalainen
                                <kimmo.kymalainen&etsi.org>
                                01 September 2009
   #                36443/tcp   <not assigned>
   m2ap             36443/sctp  M2 Application Part
   #                            Dario S. Tonesi
                                <dario.tonesi&nsn.com>
```

		07 February 2011
#	36444/tcp	<not assigned>
m3ap	36444/sctp	M3 Application Part
#		Dario S. Tonesi <dario.tonesi&nsn.com> 07 February 2011

Table 3: IANA layer 4 port collisions between SCTP and TCP

Instead of imposing the transport protocol (UDP/TCP/SCTP/etc.) in the scope of the "options application-table" Options Template for all applications (on top of having the transport protocol as key-field in the Flow Record definition), we define that the L4 application is always TCP related, by convention. So, whenever the Collector has a conflict in looking up IANA, it would choose the TCP choice. As a result, the UDP L4 applications from Table 2 and the SCTP L4 applications from table 3 are assigned in the Cisco L7 Application Tag range (ie, under Classification Engine ID 13):

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

5. Grouping the Applications with the Attributes

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

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

Name	Description
Category	An attribute that provides a first level categorization for each application tag. Examples include: browsing, email, file-sharing, gaming, instant messaging, voice-and-video, etc... The category attribute is encoded by the ApplicationCategoryName Information Element.

Application-Group	An attribute that groups multiple application tags that belong to the same networking application. For example, the ftp-group contain the ftp-data (port 20), ftp (port 20), ni-ftp (port 47), sftp (port 115), bftp (port 152), ftp-agent(port 574), ftps-data (port 989). The application-group attribute is encoded by the ApplicationGroupName Information Element.
P2P-Technology	Specifies if the application tag is based on peer-to-peer technology. The P2P-technology attribute is encoded by the p2pTechnology Information Element.
Tunnel-Technology	Specifies if the application tag is used as a tunnel technology. The tunnel-technology attribute is encoded by the tunnelTechnology Information Element.
Encrypted	Specifies if the application tag is an encrypted networking protocol. The encrypted attribute is encoded by the encryptedTechnology Information Element.

Table 4: Existing Application Tag Static Attributes

Every application is assigned to one ApplicationCategoryName, one ApplicationGroupName, has one p2pTechnology, one tunnelTechnology, and one encryptedTechnology.

5.1. Options Template Record for the Attribute Values

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

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

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

An example is given in section 6.8. below.

6. Application Tag Examples

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

6.1. Example 1: Layer 2 Protocol

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

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

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

```
NAME    Selector ID
ppp     24
```

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

Therefore the Application Tag is encoded as:

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

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

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

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

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

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

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

6.2. Example 2: Standardized IANA Layer 3 Protocol

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

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

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

Decimal	Keyword	Protocol	Reference

```
   1          ICMP       Internet Control Message    [RFC792]
```

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

Therefore the Application Tag is encoded as:

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

So the Application Tag has the value of 257. Instead of representing the Application Tag in hexadecimal format, the format '1...1' is used for simplicity in the examples below.

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

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

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

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

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

6.3. Example 3: Cisco Systems Proprietary Layer 3 Protocol

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

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

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

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

Protocol	Protocol Id
foo	90

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

Therefore the Application Tag is encoded as:

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

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

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

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

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

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

Along with this Flow Record, a new Options Template Record would be exported, as shown in Section 6.7.

6.4. Example 4: Standardized IANA Layer 4 Port

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

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

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

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

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

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

Therefore the Application Tag is encoded as:

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

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

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

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

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

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

6.5. Example 4: Layer 7 Application

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

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

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

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

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

Therefore the Application Tag is encoded as:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      13       |                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             10000                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

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

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

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

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

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

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

Along with this Flow Record, a new Options Template Record would be exported, as shown in Section 6.7.

6.6. Example: port Obfuscation

For example, a HTTP server might run a TCP port 23 (assigned to telnet in [IANA-PORTS]). If the Metering Process is capable of detecting HTTP in the same case, the Application Tag representation must contain HTTP. However, if the reporting application wants to determine whether or the default HTTP port

80 or 8080 was used, it must export the destination port (destinationTransportPort at [IANA-IPFIX]) in the corresponding NetFlow record.

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

Therefore the Application Tag is encoded as:

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

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

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

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

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

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

6.7. Example: Application Mapping Options Template

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

The Options Template Record contains the following Information Elements:

1. Scope = applicationTag.

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

2. applicationName.

3. applicationDescription.

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

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

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

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

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

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

6.8. Example: Attributes Values Options Template Record

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

The Options Template Record would contain the following Information Elements:

1. Scope = applicationTag.

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

2. applicationCategoryName.

3. applicationGroupName

4. p2pTechnology

5. tunnelTechnology

6. encryptedTechnology

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

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

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

A flow of 123456 bytes exists from sourceIPv4Address 1.1.1.1 to destinationIPv4address 2.2.2.2 with a DSCP value of 0 and an applicationTag of '12...90', which maps to the "foo" application. This application can be characterized by the relevant attributes values.

7. IANA Considerations

This document specifies 9 new IPFIX Information Elements: the applicationDescription, applicationTag, applicationName, classificationEngineId, applicationCategoryName, applicationGroupName, p2pTechnology, tunnelTechnology, and encryptedTechnology.

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

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

7.1. applicationDescription

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

7.2. applicationTag

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

7.3. applicationName

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

7.4. classificationEngineId

Name: classificationEngineId
Description:
  Specifies the classification engine according to Table 1 in [EDITORS NOTE: this document].
Abstract Data Type: unsigned8
Data Type Semantics: identifier

ElementId: 101
Status: current

7.5. applicationCategoryName

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

7.6. applicationGroupName

Name: applicationGroupName
Description:
An attribute that groups multiple Application Tags that belong to the same networking application
Abstract Data Type: string
Data Type Semantics:
ElementId: <to be assigned>
Status: current

7.7. p2pTechnology

Name: p2pTechnology
Description:
Specifies if the Application Tag is based on peer-to-peer technology. Possible values are: "yes", "no", and "unassigned"
Abstract Data Type: string
Data Type Semantics:
ElementId: 288
Status: current

7.8. tunnelTechnology

Name: tunnelTechnology
Description:
Specifies if the application tag is used as a tunnel technology.
Possible values are: "yes", "no", and "unassigned"

Abstract Data Type: string
Data Type Semantics:
ElementId: 289
Status: current

7.9. encryptedTechnology

Name: encryptedTechnology
Description:
Specifies if the application tag is an encrypted networking protocol. Possible values are: "yes", "no", and "unassigned"
Abstract Data Type: string
Data Type Semantics:
ElementId: 290
Status: current

8. Security Considerations

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

9. References

9.1. Normative References

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

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

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

9.2. Informative References

[RFC792] J. Postel, Internet Control Message Protocol, RFC 792, September 1981.

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

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

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

[RFC5471] Schmoll, C., Aitken, P., and B. Claise, "Guidelines for IP Flow Information Export (IPFIX) Testing", RFC 5471, March 2009.

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

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

[RFC6313] Claise, B., Dhandapani, G. Aitken, P., and S. Yates, "Export of Structured Data in IP Flow Information Export (IPFIX)", RFC6313, July 20111

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

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

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

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

10. Acknowledgement

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

11. Authors' Addresses

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

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

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

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

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

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

Appendix A. Additions to XML Specification of IPFIX Information Elements

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

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


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

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

  <field name="applicationName"
         dataType="string"
         group="application"
         elementId="96" applicability="all" status="current">
    <description>
      <paragraph>
```

```
            Specifies the name of an application.
          </paragraph>
        </description>
      </field>

      <field name="classificationEngineId"
             dataType="unsigned8"
             group="application"
             dataTypeSemantics="identifer"
             elementId="101" applicability="all" status="current">
        <description>
          <paragraph>
            Specifies the classification engine according to Table
   1 in [EDITORS NOTE: this document].
          </paragraph>
        </description>
      </field>

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

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

      <field name="p2pTechnology"
             dataType="string"
             group="application"
```

```
            elementId="288" applicability="all" status="current">
        <description>
          <paragraph>
                Specifies if the Application Tag is based on peer-
                to-peer technology. Possible values are: "yes",
                "no", and "unassigned".
          </paragraph>
        </description>
      </field>

      <field name="tunnelTechnology"
             dataType="string"
             group="application"
             elementId="289" applicability="all" status="current">
        <description>
          <paragraph>
                  Specifies if the application tag is used as a
                  tunnel technology. Possible values are: "yes",
                  "no", and "unassigned".
          </paragraph>
        </description>
      </field>

      <field name="encryptedTechnology"
             dataType="string"
             group="application"
             elementId="290" applicability="all" status="current">
        <description>
          <paragraph>
                  Specifies if the application tag is an encrypted
                  networking protocol. Possible values are: "yes",
                  "no", and "unassigned".
          </paragraph>
        </description>
      </field>
```

Network Working Group J. Quittek
Internet Draft NEC
Obsoletes: 5102 S. Bryant
Category: Standards Track B. Claise
Expires: April 30, 2012 P. Aitken
Cisco Systems, Inc.
J. Meyer
PayPal
October 28, 2011

Information Model for IP Flow Information eXport (IPFIX)
draft-claise-ipfix-information-model-rfc5102bis-01.txt

Abstract

This memo defines an information model for the IP Flow Information eXport (IPFIX) protocol. It is used by the IPFIX protocol for encoding measured traffic information and information related to the traffic Observation Point, the traffic Metering Process, and the Exporting Process. Although developed for the IPFIX protocol, the model is defined in an open way that easily allows using it in other protocols, interfaces, and applications. This document obsoletes RFC 5102.

Status of This Memo

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

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

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

This Internet-Draft will expire on March 23, 2012.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust’s Legal

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

Table of Contents

DONE:
- Errata ID: 1307 (technical)
- Errata ID: 1492 (technical)
- Errata ID: 1736 (technical)
- Errata ID: 2879 (editorial)
- Errata ID: 2944, which updates 1737 (technical)
- Errata ID: 2945, which updates 1738 (technical)
- Errata ID: 2946, which updates 1739 (technical)
- Updated the reference to RFC5101bis
- Clarified the time-related IEs

TO DO:
- IPFIX Documents Overview
- Should we repeat the IEs in this RFC or should we point to IANA?
- Clarify the interaction with http://tools.ietf.org/html/draft-trammell-ipfix-ie-doctors-02
- IPFIX XML is different in the registry? Checked with Brian
- some IE definitions have a reference to RFC5101. Should they refer to RFC5101bis, which implies that the IANA registry would have to be changed? Example: exportProtocolVersion. Note that the XML must be kept in line, and that only exportProtocolVersion has been modified in XML.
- Found "to be done" in the appendix A in "<attribute name="group" type="string" use="required">

1. Introduction

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

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

The use of the term 'information model' is not fully in line with the definition of this term in [RFC3444]. The IPFIX information model does not specify relationships between Information Elements, but also it does not specify a concrete encoding of Information Elements. Besides the encoding used by the IPFIX protocol, other encodings of IPFIX Information Elements can be applied, for example, XML-based encodings.

The main part of this document is Section 5, which defines the (extensible) list of Information Elements to be transmitted by the IPFIX protocol. Section 2 defines a template for specifying IPFIX Information Elements in Section 5. Section 3 defines the set of abstract data types that are available for IPFIX Information Elements. Section 6 discusses extensibility of the IPFIX information model.

The main bodies of Sections 2, 3, and 5 were generated from XML documents. The XML-based specification of template, abstract data types, and IPFIX Information Elements can be used for automatically checking syntactical correctness of the specification of IPFIX Information Elements. It can further be used for generating IPFIX protocol implementation code that deals with processing IPFIX Information Elements. Also, code for applications that further process traffic information transmitted via the IPFIX protocol can be generated with the XML specification of IPFIX Information Elements.

For that reason, the XML document that served as a source for Section 5 and the XML schema that served as source for Sections 2 and 3 are attached to this document in Appendices A and B.

Note that although partially generated from the attached XML documents, the main body of this document is normative while the appendices are informational.

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

1.1. IPFIX Documents Overview

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

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

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

[IPFIX-CONF] specifies a data model for configuring and monitoring IPFIX and PSAMP compliant devices using the NETCONF protocol, while the [RFC5815bis] specifies a MIB module for monitoring.

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

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

2. Properties of IPFIX Protocol Information Elements

2.1. Information Elements Specification Template

Information in messages of the IPFIX protocol is modeled in terms of Information Elements of the IPFIX information model. IPFIX Information Elements are specified in Section 5. For specifying these Information Elements, a template is used that is described below.

All Information Elements specified for the IPFIX protocol either in this document or by any future extension 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.

dataType - One of the types listed in Section 3.1 of this document or in a future extension of the information model. The type space for attributes is constrained to facilitate implementation. The existing type space does however encompass most basic types used in modern programming languages, as well as some derived types (such as ipv4Address) that are common to this domain and useful to distinguish.

status - The status of the specification of this Information Element. Allowed values are 'current', 'deprecated', and 'obsolete'.

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

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

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

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

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

range - Some Information Elements may only be able to take on a restricted set of values that can be expressed as a range (e.g., 0 through 511 inclusive). If this is the case, the valid inclusive range should be specified.

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

2.2. Scope of Information Elements

By default, most Information Elements have a scope specified in their definitions.

- o The Information Elements defined in Sections 5.2 and 5.3 have a default of "a specific Metering Process" or of "a specific Exporting Process", respectively.

- o The Information Elements defined in Sections 5.4-5.11 have a scope of "a specific Flow".

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

2.3. Naming Conventions for Information Elements

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

- o Names of Information Elements should be descriptive.

- o Names of Information Elements that are not enterprise-specific MUST be unique within the IPFIX information model. Enterprise-specific Information Elements SHOULD be prefixed with a vendor name.

- o Names of Information Elements start with non-capitalized letters.

- o Composed names use capital letters for the first letter of each component (except for the first one). All other letters are non-capitalized, even for acronyms. Exceptions are made for acronyms containing non-capitalized letter, such as 'IPv4' and 'IPv6'. Examples are sourceMacAddress and destinationIPv4Address.

- o Middleboxes [RFC3234] may change Flow properties, such as the

Differentiated Service Code Point (DSCP) value or the source IP address. If an IPFIX Observation Point is located in the path of a Flow before one or more middleboxes that potentially modify packets of the Flow, then it may be desirable to also report Flow properties after the modification performed by the middleboxes. An example is an Observation Point before a packet marker changing a packet’s IPv4 Type of Service (TOS) field that is encoded in Information Element ipClassOfService. Then the value observed and reported by Information Element ipClassOfService is valid at the Observation Point, but not after the packet passed the packet marker. For reporting the change value of the TOS field, the IPFIX information model uses Information Elements that have a name prefix "post", for example, "postIpClassOfService". Information Elements with prefix "post" report on Flow properties that are not necessarily observed at the Observation Point, but which are obtained within the Flow’s Observation Domain by other means considered to be sufficiently reliable, for example, by analyzing the packet marker’s marking tables.

3. Type Space

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

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

3.1. Abstract Data Types

This section describes the set of valid abstract data types of the IPFIX information model. Note that further abstract data types may be specified by future extensions of the IPFIX information model.

3.1.1. unsigned8

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

3.1.2. unsigned16

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

3.1.3. unsigned32

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

3.1.4. unsigned64

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

3.1.5. signed8

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

3.1.6. signed16

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

3.1.7. signed32

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

3.1.8. signed64

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

3.1.9. float32

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

3.1.10. float64

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

3.1.11. boolean

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

3.1.12. macAddress

The type "macAddress" represents a string of 6 octets.

3.1.13. octetArray

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

3.1.14. string

The type "string" represents a finite-length string of valid characters from the Unicode character encoding set [ISO.10646-1.1993]. Unicode allows for ASCII [ISO.646.1991] and many other international character sets to be used.

3.1.15. dateTimeSeconds

The type "dateTimeSeconds" represents a time value in units of seconds since the UNIX epoch, 1 January 1970 at 00:00 coordinated universal time (UTC), excluding leap seconds.

3.1.16. dateTimeMilliseconds

The type "dateTimeSeconds" represents a time value in units of milliseconds since the UNIX epoch, 1 January 1970 at 00:00 coordinated universal time (UTC), excluding leap seconds.

3.1.17. dateTimeMicroseconds

The type "dateTimeMicroseconds" represents a time value with microsecond precision according to the NTP Timestamp format as defined in section 6 of [RFC5905]. This field is made up of two unsigned 32-bit integers, Seconds and Fraction. The Seconds field is the number of seconds since the NTP epoch, 1 January 1900 at 00:00 UTC. The Fraction field is the fractional number of seconds in units of 1/(2^32) seconds (approximately 233 picoseconds).

3.1.18. dateTimeNanoseconds

The type "dateTimeMicroseconds" represents a time value with nanosecond precision according to the NTP Timestamp format as defined in section 6 of [RFC5905]. This field is made up of two unsigned 32-bit integers, Seconds and Fraction. The Seconds field is the number of seconds since the NTP epoch, 1 January 1900 at 00:00 UTC. The Fraction field is the fractional number of seconds in units of 1/(2^32) seconds (approximately 233 picoseconds).

3.1.19. ipv4Address

The type "ipv4Address" represents a value of an IPv4 address.

3.1.20. ipv6Address

The type "ipv6Address" represents a value of an IPv6 address.

3.2. Data Type Semantics

This section describes the set of valid data type semantics of the IPFIX information model. Note that further data type semantics may be specified by future extensions of the IPFIX information model.

3.2.1. quantity

A quantity value represents a discrete 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. If no semantic qualifier is given, the Information Elements that have an integral data type should behave as a quantity.

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 RFC 2578 [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 its value is exported.

3.2.4. identifier

An integral value that serves as an identifier. Specifically, mathematical operations on two identifiers (aside from the equality operation) are meaningless. For example, Autonomous System ID 1 * Autonomous System ID 2 is meaningless.

3.2.5. flags

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

4. Information Element Identifiers

All Information Elements defined in Section 5 of this document or in future extensions of the IPFIX information model have their identifiers assigned by IANA. Their identifiers can be retrieved at http://www.iana.org/assignments/ipfix.

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].

Range of IANA-assigned Information Element identifiers	Description
0	Reserved.
1-127	Information Element identifiers compatible with NetFlow version 9 field types [RFC3954].
128-32767	Further Information Element identifiers.

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.

Still, Collecting Processes can distinguish these Information Elements because the Information Element identifier is coupled with an enterprise identifier.

Enterprise identifiers MUST be registered as SMI network management private enterprise code numbers with IANA. The registry can be found at http://www.iana.org/assignments/enterprise-numbers.

The following list gives an overview of the Information Element identifiers that are specified in Section 5 and are compatible with field types used by NetFlow version 9 [RFC3954].

ID	Name	ID	Name
1	octetDeltaCount	43	RESERVED
2	packetDeltaCount	44	sourceIPv4Prefix
3	RESERVED	45	destinationIPv4Prefix
4	protocolIdentifier	46	mplsTopLabelType
5	ipClassOfService	47	mplsTopLabelIPv4Address
6	tcpControlBits	48-51	RESERVED
7	sourceTransportPort	52	minimumTTL
8	sourceIPv4Address	53	maximumTTL
9	sourceIPv4PrefixLength	54	fragmentIdentification
10	ingressInterface	55	postIpClassOfService
11	destinationTransportPort	56	sourceMacAddress
12	destinationIPv4Address	57	postDestinationMacAddress
13	destinationIPv4PrefixLength	58	vlanId
14	egressInterface	59	postVlanId
15	ipNextHopIPv4Address	60	ipVersion
16	bgpSourceAsNumber	61	flowDirection
17	bgpDestinationAsNumber	62	ipNextHopIPv6Address
18	bgpNexthopIPv4Address	63	bgpNexthopIPv6Address
19	postMCastPacketDeltaCount	64	ipv6ExtensionHeaders
20	postMCastOctetDeltaCount	65-69	RESERVED
21	flowEndSysUpTime	70	mplsTopLabelStackSection
22	flowStartSysUpTime	71	mplsLabelStackSection2
23	postOctetDeltaCount	72	mplsLabelStackSection3
24	postPacketDeltaCount	73	mplsLabelStackSection4
25	minimumIpTotalLength	74	mplsLabelStackSection5
26	maximumIpTotalLength	75	mplsLabelStackSection6
27	sourceIPv6Address	76	mplsLabelStackSection7
28	destinationIPv6Address	77	mplsLabelStackSection8
29	sourceIPv6PrefixLength	78	mplsLabelStackSection9
30	destinationIPv6PrefixLength	79	mplsLabelStackSection10
31	flowLabelIPv6	80	destinationMacAddress
32	icmpTypeCodeIPv4	81	postSourceMacAddress
33	igmpType	82-84	RESERVED
34	RESERVED	85	octetTotalCount
35	RESERVED	86	packetTotalCount
36	flowActiveTimeout	87	RESERVED
37	flowIdleTimeout	88	fragmentOffset
38	RESERVED	89	RESERVED
39	RESERVED	90	mplsVpnRouteDistinguisher
40	exportedOctetTotalCount	91-127	RESERVED
41	exportedMessageTotalCount		
42	exportedFlowRecordTotalCount		

The following list gives an overview of the Information Element identifiers that are specified in Section 5 and extends the list of Information Element identifiers specified already in [RFC3954].

ID	Name	ID	Name
128	bgpNextAdjacentAsNumber	169	destinationIPv6Prefix
129	bgpPrevAdjacentAsNumber	170	sourceIPv6Prefix
130	exporterIPv4Address	171	postOctetTotalCount
131	exporterIPv6Address	172	postPacketTotalCount
132	droppedOctetDeltaCount	173	flowKeyIndicator
133	droppedPacketDeltaCount	174	postMCastPacketTotalCount
134	droppedOctetTotalCount	175	postMCastOctetTotalCount
135	droppedPacketTotalCount	176	icmpTypeIPv4
136	flowEndReason	177	icmpCodeIPv4
137	commonPropertiesId	178	icmpTypeIPv6
138	observationPointId	179	icmpCodeIPv6
139	icmpTypeCodeIPv6	180	udpSourcePort
140	mplsTopLabelIPv6Address	181	udpDestinationPort
141	lineCardId	182	tcpSourcePort
142	portId	183	tcpDestinationPort
143	meteringProcessId	184	tcpSequenceNumber
144	exportingProcessId	185	tcpAcknowledgementNumber
145	templateId	186	tcpWindowSize
146	wlanChannelId	187	tcpUrgentPointer
147	wlanSSID	188	tcpHeaderLength
148	flowId	189	ipHeaderLength
149	observationDomainId	190	totalLengthIPv4
150	flowStartSeconds	191	payloadLengthIPv6
151	flowEndSeconds	192	ipTTL
152	flowStartMilliseconds	193	nextHeaderIPv6
153	flowEndMilliseconds	194	mplsPayloadLength
154	flowStartMicroseconds	195	ipDiffServCodePoint
155	flowEndMicroseconds	196	ipPrecedence
156	flowStartNanoseconds	197	fragmentFlags
157	flowEndNanoseconds	198	octetDeltaSumOfSquares
158	flowStartDeltaMicroseconds	199	octetTotalSumOfSquares
159	flowEndDeltaMicroseconds	200	mplsTopLabelTTL
160	systemInitTimeMilliseconds	201	mplsLabelStackLength
161	flowDurationMilliseconds	202	mplsLabelStackDepth
162	flowDurationMicroseconds	203	mplsTopLabelExp
163	observedFlowTotalCount	204	ipPayloadLength
164	ignoredPacketTotalCount	205	udpMessageLength
165	ignoredOctetTotalCount	206	isMulticast
166	notSentFlowTotalCount	207	ipv4IHL
167	notSentPacketTotalCount	208	ipv4Options
168	notSentOctetTotalCount	209	tcpOptions

ID	Name	ID	Name
210	paddingOctets	218	tcpSynTotalCount
211	collectorIPv4Address	219	tcpFinTotalCount
212	collectorIPv6Address	220	tcpRstTotalCount
213	exportInterface	221	tcpPshTotalCount
214	exportProtocolVersion	222	tcpAckTotalCount
215	exportTransportProtocol	223	tcpUrgTotalCount
216	collectorTransportPort	224	ipTotalLength
217	exporterTransportPort	237	postMplsTopLabelExp
		238	tcpWindowScale

5. Information Elements

This section describes the Information Elements of the IPFIX information model. The elements are grouped into 12 groups according to their semantics and their applicability:

1. Identifiers
2. Metering and Exporting Process Configuration
3. Metering and Exporting Process Statistics
4. IP Header Fields
5. Transport Header Fields
6. Sub-IP Header Fields
7. Derived Packet Properties
8. Min/Max Flow Properties
9. Flow Timestamps
10. Per-Flow Counters
11. Miscellaneous Flow Properties
12. Padding

The Information Elements that are derived from fields of packets or from packet treatment, such as the Information Elements in groups 4-7, can typically serve as Flow Keys used for mapping packets to Flows.

If they do not serve as Flow Keys, their value may change from packet to packet within a single Flow. For Information Elements with values that are derived from fields of packets or from packet treatment and for which the value may change from packet to packet within a single Flow, the IPFIX information model defines that their value is determined by the first packet observed for the corresponding Flow, unless the description of the Information Element explicitly specifies a different semantics. This simple rule allows writing all

Information Elements related to header fields once when the first packet of the Flow is observed. For further observed packets of the same Flow, only Flow properties that depend on more than one packet, such as the Information Elements in groups 8-11, need to be updated.

Information Elements with a name having the "post" prefix, for example, "postIpClassOfService", do not report properties that were actually observed at the Observation Point, but retrieved by other means within the Observation Domain. These Information Elements can be used if there are middlebox functions within the Observation Domain changing Flow properties after packets passed the Observation Point.

Information Elements in this section use the reference property to reference [RFC0768], [RFC0791], [RFC0792], [RFC0793], [RFC1108], [RFC1112], [RFC1191], [RFC1323], [RFC1385], [RFC1812], [RFC1930], [RFC2113], [RFC2119], [RFC2460], [RFC2675], [RFC2863], [RFC3031], [RFC3032], [RFC3193], [RFC3234], [RFC3260], [RFC3270], [RFC3376], [RFC3954], [RFC4271], [RFC4291], [RFC4302], [RFC4303], [RFC4364], [RFC4382], [RFC4443], [RFC4960], [RFC5036], [IEEE.802-11.1999], [IEEE.802-1Q.2003], and [IEEE.802-3.2002].

5.1. Identifiers

Information Elements grouped in the table below are identifying components of the IPFIX architecture, of an IPFIX Device, or of the IPFIX protocol. All of them have an integral abstract data type and data type semantics "identifier" as described in Section 3.2.4.

Typically, some of them are used for limiting scopes of other Information Elements. However, other Information Elements MAY be used for limiting scopes. Note also that all Information Elements listed below MAY be used for other purposes than limiting scopes.

ID	Name	ID	Name
141	lineCardId	148	flowId
142	portId	145	templateId
10	ingressInterface	149	observationDomainId
14	egressInterface	138	observationPointId
143	meteringProcessId	137	commonPropertiesId
144	exportingProcessId		

5.1.1. lineCardId

Description:
An identifier of a line card that is unique per IPFIX Device hosting an Observation Point. Typically, this Information Element is used for limiting the scope of other Information Elements.
Abstract Data Type: unsigned32
Data Type Semantics: identifier
ElementId: 141
Status: current

5.1.2. portId

Description:
An identifier of a line port that is unique per IPFIX Device hosting an Observation Point. Typically, this Information Element is used for limiting the scope of other Information Elements.
Abstract Data Type: unsigned32
Data Type Semantics: identifier
ElementId: 142
Status: current

5.1.3. ingressInterface

Description:
The index of the IP interface where packets of this Flow are being received. The value matches the value of managed object 'ifIndex' as defined in RFC 2863. Note that ifIndex values are not assigned statically to an interface and that the interfaces may be renumbered every time the device's management system is re-initialized, as specified in RFC 2863.
Abstract Data Type: unsigned32
Data Type Semantics: identifier
ElementId: 10
Status: current
Reference:
See RFC 2863 for the definition of the ifIndex object.

5.1.4. egressInterface

Description:
The index of the IP interface where packets of this Flow are being sent. The value matches the value of managed object 'ifIndex' as defined in RFC 2863. Note that ifIndex values are not assigned statically to an interface and that the interfaces may be renumbered every time the device's management system is re-initialized, as specified in RFC 2863.
Abstract Data Type: unsigned32
Data Type Semantics: identifier
ElementId: 14
Status: current
Reference:
See RFC 2863 for the definition of the ifIndex object.

5.1.5. meteringProcessId

Description:
An identifier of a Metering 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 are typically assigned dynamically. The Metering Process may be re-started with a different ID.
Abstract Data Type: unsigned32
Data Type Semantics: identifier
ElementId: 143
Status: current

5.1.6. exportingProcessId

Description:
An identifier of an Exporting 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 are typically assigned dynamically. The Exporting Process may be re-started with a different ID.
Abstract Data Type: unsigned32
Data Type Semantics: identifier
ElementId: 144
Status: current

5.1.7. flowId

Description:
An identifier of a Flow that is unique within an Observation Domain. This Information Element can be used to distinguish between different Flows if Flow Keys such as IP addresses and port numbers are not reported or are reported in separate records.
Abstract Data Type: unsigned64
Data Type Semantics: identifier
ElementId: 148
Status: current

5.1.8. templateId

Description:
An identifier of a Template that is locally unique within a combination of a Transport session and an Observation Domain. Template IDs 0-255 are reserved for Template Sets, Options Template Sets, and other reserved Sets yet to be created. Template IDs of Data Sets are numbered from 256 to 65535. Typically, this Information Element is used for limiting the scope of other Information Elements. Note that after a re-start of the Exporting Process Template identifiers may be re-assigned.
Abstract Data Type: unsigned16
Data Type Semantics: identifier
ElementId: 145
Status: current

5.1.9. observationDomainId

Description:
An identifier of an Observation Domain that is locally unique to an Exporting Process. The Exporting Process uses the Observation Domain ID to uniquely identify to the Collecting Process the Observation Domain where Flows were metered. It is RECOMMENDED that this identifier is also unique per IPFIX Device. A value of 0 indicates that no specific Observation Domain is identified by this Information Element. Typically, this Information Element is used for limiting the scope of other Information Elements.
Abstract Data Type: unsigned32
Data Type Semantics: identifier
ElementId: 149
Status: current

5.1.10. observationPointId

Description:
An identifier of an Observation Point that is unique per Observation Domain. It is RECOMMENDED that this identifier is also unique per IPFIX Device. Typically, this Information Element is used for limiting the scope of other Information Elements.

Abstract Data Type: unsigned32
Data Type Semantics: identifier
ElementId: 138
Status: current

5.1.11. commonPropertiesId

Description:
An identifier of a set of common properties that is unique per Observation Domain and Transport Session. Typically, this Information Element is used to link to information reported in separate Data Records.

Abstract Data Type: unsigned64
Data Type Semantics: identifier
ElementId: 137
Status: current

5.2. Metering and Exporting Process Configuration

Information Elements in this section describe the configuration of the Metering Process or the Exporting Process. The set of these Information Elements is listed in the table below.

ID	Name	ID	Name
130	exporterIPv4Address	213	exportInterface
131	exporterIPv6Address	214	exportProtocolVersion
217	exporterTransportPort	215	exportTransportProtocol
211	collectorIPv4Address	216	collectorTransportPort
212	collectorIPv6Address	173	flowKeyIndicator

5.2.1. exporterIPv4Address

Description:
The IPv4 address used by the Exporting Process. This is used by the Collector to identify the Exporter in cases where the identity of the Exporter may have been obscured by the use of a proxy.
Abstract Data Type: ipv4Address
Data Type Semantics: identifier
ElementId: 130
Status: current

5.2.2. exporterIPv6Address

Description:
The IPv6 address used by the Exporting Process. This is used by the Collector to identify the Exporter in cases where the identity of the Exporter may have been obscured by the use of a proxy.
Abstract Data Type: ipv6Address
Data Type Semantics: identifier
ElementId: 131
Status: current

5.2.3. exporterTransportPort

Description:
The source port identifier from which the Exporting Process sends Flow information. For the transport protocols UDP, TCP, and SCTP, this is the source port number. This field MAY also be used for future transport protocols that have 16-bit source port identifiers. This field may be useful for distinguishing multiple Exporting Processes that use the same IP address.
Abstract Data Type: unsigned16
Data Type Semantics: identifier
ElementId: 217
Status: current
Reference:
See RFC 768 for the definition of the UDP source port field. See RFC 793 for the definition of the TCP source port field. See RFC 4960 for the definition of SCTP. Additional information on defined UDP and TCP port numbers can be found at http://www.iana.org/assignments/port-numbers.

5.2.4. collectorIPv4Address

Description:
An IPv4 address to which the Exporting Process sends Flow information.
Abstract Data Type: ipv4Address
Data Type Semantics: identifier
ElementId: 211
Status: current

5.2.5. collectorIPv6Address

Description:
An IPv6 address to which the Exporting Process sends Flow information.
Abstract Data Type: ipv6Address
Data Type Semantics: identifier
ElementId: 212
Status: current

5.2.6. exportInterface

Description:
The index of the interface from which IPFIX Messages sent by the Exporting Process to a Collector leave the IPFIX Device. The value matches the value of managed object 'ifIndex' as defined in RFC 2863. Note that ifIndex values are not assigned statically to an interface and that the interfaces may be renumbered every time the device's management system is re-initialized, as specified in RFC 2863.
Abstract Data Type: unsigned32
Data Type Semantics: identifier
ElementId: 213
Status: current
Reference:
See RFC 2863 for the definition of the ifIndex object.

5.2.7. exportProtocolVersion

Description:
The protocol version used by the Exporting Process for sending Flow information. The protocol version is given by the value of the Version Number field in the Message Header. The protocol version is 10 for IPFIX and 9 for NetFlow version 9. A value of 0 indicates that no export protocol is in use.
Abstract Data Type: unsigned8
Data Type Semantics: identifier
ElementId: 214
Status: current
Reference:
See the IPFIX protocol specification [RFC5101] for the definition of the IPFIX Message Header.
See RFC 3954 for the definition of the NetFlow version 9 message header.

5.2.8. exportTransportProtocol

Description:
The value of the protocol number used by the Exporting Process for sending Flow information. The protocol number identifies the IP packet payload type. Protocol numbers are defined in the IANA Protocol Numbers registry.
In Internet Protocol version 4 (IPv4), this is carried in the Protocol field. In Internet Protocol version 6 (IPv6), this is carried in the Next Header field in the last extension header of the packet.
Abstract Data Type: unsigned8
Data Type Semantics: identifier
ElementId: 215
Status: current
Reference:
See RFC 791 for the specification of the IPv4 protocol field. See RFC 2460 for the specification of the IPv6 protocol field. See the list of protocol numbers assigned by IANA at http://www.iana.org/assignments/protocol-numbers.

5.2.9. collectorTransportPort

Description:
The destination port identifier to which the Exporting Process sends Flow information. For the transport protocols UDP, TCP, and SCTP, this is the destination port number. This field MAY also be used for future transport protocols that have 16-bit source port identifiers.

Abstract Data Type: unsigned16
Data Type Semantics: identifier
ElementId: 216
Status: current
Reference:
See RFC 768 for the definition of the UDP destination port field.
See RFC 793 for the definition of the TCP destination port field.
See RFC 4960 for the definition of SCTP.
Additional information on defined UDP and TCP port numbers can be found at http://www.iana.org/assignments/port-numbers.

5.2.10. flowKeyIndicator

Description:
This set of bit fields is used for marking the Information Elements of a Data Record that serve as Flow Key. Each bit represents an Information Element in the Data Record with the n-th bit representing the n-th Information Element. A bit set to value 1 indicates that the corresponding Information Element is a Flow Key of the reported Flow. A bit set to value 0 indicates that this is not the case.
If the Data Record contains more than 64 Information Elements, the corresponding Template SHOULD be designed such that all Flow Keys are among the first 64 Information Elements, because the flowKeyIndicator only contains 64 bits. If the Data Record contains less than 64 Information Elements, then the bits in the flowKeyIndicator for which no corresponding Information Element exists MUST have the value 0.

Abstract Data Type: unsigned64
Data Type Semantics: flags
ElementId: 173
Status: current

5.3. Metering and Exporting Process Statistics

Information Elements in this section describe statistics of the Metering Process and/or the Exporting Process. The set of these Information Elements is listed in the table below.

ID	Name	ID	Name
41	exportedMessageTotalCount	165	ignoredOctetTotalCount
40	exportedOctetTotalCount	166	notSentFlowTotalCount
42	exportedFlowRecordTotalCount	167	notSentPacketTotalCount
163	observedFlowTotalCount	168	notSentOctetTotalCount
164	ignoredPacketTotalCount		

5.3.1. exportedMessageTotalCount

Description:
The total number of IPFIX Messages that the Exporting Process has sent since the Exporting Process (re-)initialization to a particular Collecting Process. The reported number excludes the IPFIX Message that carries the counter value. If this Information Element is sent to a particular Collecting Process, then by default it specifies the number of IPFIX Messages sent to this Collecting Process.
Abstract Data Type: unsigned64
Data Type Semantics: totalCounter
ElementId: 41
Status: current
Units: messages

5.3.2. exportedOctetTotalCount

Description:
The total number of octets that the Exporting Process has sent since the Exporting Process (re-)initialization to a particular Collecting Process. The value of this Information Element is calculated by summing up the IPFIX Message Header length values of all IPFIX Messages that were successfully sent to the Collecting Process. The reported number excludes octets in the IPFIX Message that carries the counter value. If this Information Element is sent to a particular Collecting Process, then by default it specifies the number of octets sent to this Collecting Process.
Abstract Data Type: unsigned64
Data Type Semantics: totalCounter
ElementId: 40
Status: current

Units: octets

5.3.3. exportedFlowRecordTotalCount

Description:
The total number of Flow Records that the Exporting Process has sent as Data Records since the Exporting Process (re-)initialization to a particular Collecting Process. The reported number excludes Flow Records in the IPFIX Message that carries the counter value. If this Information Element is sent to a particular Collecting Process, then by default it specifies the number of Flow Records sent to this process.
Abstract Data Type: unsigned64
Data Type Semantics: totalCounter
ElementId: 42
Status: current
Units: flows

5.3.4. observedFlowTotalCount

Description:
The total number of Flows observed in the Observation Domain since the Metering Process (re-)initialization for this Observation Point.
Abstract Data Type: unsigned64
Data Type Semantics: totalCounter
ElementId: 163
Status: current
Units: flows

5.3.5. ignoredPacketTotalCount

Description:
The total number of observed IP packets that the Metering Process did not process since the (re-)initialization of the Metering Process.
Abstract Data Type: unsigned64
Data Type Semantics: totalCounter
ElementId: 164
Status: current
Units: packets

5.3.6. ignoredOctetTotalCount

Description:
The total number of octets in observed IP packets (including the IP header) that the Metering Process did not process since the (re-)initialization of the Metering Process.
Abstract Data Type: unsigned64
Data Type Semantics: totalCounter
ElementId: 165
Status: current
Units: octets

5.3.7. notSentFlowTotalCount

Description:
The total number of Flow Records that were generated by the Metering Process and dropped by the Metering Process or by the Exporting Process instead of being sent to the Collecting Process. There are several potential reasons for this including resource shortage and special Flow export policies.
Abstract Data Type: unsigned64
Data Type Semantics: totalCounter
ElementId: 166
Status: current
Units: flows

5.3.8. notSentPacketTotalCount

Description:
The total number of packets in Flow Records that were generated by the Metering Process and dropped by the Metering Process or by the Exporting Process instead of being sent to the Collecting Process. There are several potential reasons for this including resource shortage and special Flow export policies.
Abstract Data Type: unsigned64
Data Type Semantics: totalCounter
ElementId: 167
Status: current
Units: packets

5.3.9. notSentOctetTotalCount

Description:
The total number of octets in packets in Flow Records that were generated by the Metering Process and dropped by the Metering Process or by the Exporting Process instead of being sent to the Collecting Process. There are several potential reasons for this including resource shortage and special Flow export policies.
Abstract Data Type: unsigned64
Data Type Semantics: totalCounter
ElementId: 168
Status: current
Units: octets

5.4. IP Header Fields

Information Elements in this section indicate values of IP header fields or are derived from IP header field values in combination with further information.

ID	Name	ID	Name
60	ipVersion	193	nextHeaderIPv6
8	sourceIPv4Address	195	ipDiffServCodePoint
27	sourceIPv6Address	196	ipPrecedence
9	sourceIPv4PrefixLength	5	ipClassOfService
29	sourceIPv6PrefixLength	55	postIpClassOfService
44	sourceIPv4Prefix	31	flowLabelIPv6
170	sourceIPv6Prefix	206	isMulticast
12	destinationIPv4Address	54	fragmentIdentification
28	destinationIPv6Address	88	fragmentOffset
13	destinationIPv4PrefixLength	197	fragmentFlags
30	destinationIPv6PrefixLength	189	ipHeaderLength
45	destinationIPv4Prefix	207	ipv4IHL
169	destinationIPv6Prefix	190	totalLengthIPv4
192	ipTTL	224	ipTotalLength
4	protocolIdentifier	191	payloadLengthIPv6

5.4.1. ipVersion

Description:
The IP version field in the IP packet header.
Abstract Data Type: unsigned8
Data Type Semantics: identifier
ElementId: 60
Status: current

Reference:
See RFC 791 for the definition of the version field in the IPv4 packet header. See RFC 2460 for the definition of the version field in the IPv6 packet header. Additional information on defined version numbers can be found at http://www.iana.org/assignments/version-numbers.

5.4.2. sourceIPv4Address

Description:
The IPv4 source address in the IP packet header.
Abstract Data Type: ipv4Address
Data Type Semantics: identifier
ElementId: 8
Status: current
Reference:
See RFC 791 for the definition of the IPv4 source address field.

5.4.3. sourceIPv6Address

Description:
The IPv6 source address in the IP packet header.
Abstract Data Type: ipv6Address
Data Type Semantics: identifier
ElementId: 27
Status: current
Reference:
See RFC 2460 for the definition of the Source Address field in the IPv6 header.

5.4.4. sourceIPv4PrefixLength

Description:
The number of contiguous bits that are relevant in the sourceIPv4Prefix Information Element.
Abstract Data Type: unsigned8
ElementId: 9
Status: current
Units: bits
Range: The valid range is 0-32.

5.4.5. sourceIPv6PrefixLength

Description:
The number of contiguous bits that are relevant in the sourceIPv6Prefix Information Element.
Abstract Data Type: unsigned8
ElementId: 29
Status: current
Units: bits
Range: The valid range is 0-128.

5.4.6. sourceIPv4Prefix

Description:
IPv4 source address prefix.
Abstract Data Type: ipv4Address
ElementId: 44
Status: current

5.4.7. sourceIPv6Prefix

Description:
IPv6 source address prefix.
Abstract Data Type: ipv6Address
ElementId: 170
Status: current

5.4.8. destinationIPv4Address

Description:
The IPv4 destination address in the IP packet header.
Abstract Data Type: ipv4Address
Data Type Semantics: identifier
ElementId: 12
Status: current
Reference:
See RFC 791 for the definition of the IPv4 destination address field.

5.4.9. destinationIPv6Address

Description:
The IPv6 destination address in the IP packet header.
Abstract Data Type: ipv6Address
Data Type Semantics: identifier
ElementId: 28
Status: current
Reference:
See RFC 2460 for the definition of the Destination Address field in the IPv6 header.

5.4.10. destinationIPv4PrefixLength

Description:
The number of contiguous bits that are relevant in the destinationIPv4Prefix Information Element.
Abstract Data Type: unsigned8
ElementId: 13
Status: current
Units: bits
Range: The valid range is 0-32.

5.4.11. destinationIPv6PrefixLength

Description:
The number of contiguous bits that are relevant in the destinationIPv6Prefix Information Element.
Abstract Data Type: unsigned8
ElementId: 30
Status: current
Units: bits
Range: The valid range is 0-128.

5.4.12. destinationIPv4Prefix

Description:
IPv4 destination address prefix.
Abstract Data Type: ipv4Address
ElementId: 45
Status: current

5.4.13. destinationIPv6Prefix

Description:
IPv6 destination address prefix.
Abstract Data Type: ipv6Address
ElementId: 169
Status: current

5.4.14. ipTTL

Description:
For IPv4, the value of the Information Element matches the value of the Time to Live (TTL) field in the IPv4 packet header. For IPv6, the value of the Information Element matches the value of the Hop Limit field in the IPv6 packet header.
Abstract Data Type: unsigned8
ElementId: 192
Status: current
Units: hops
Reference:
See RFC 791 for the definition of the IPv4 Time to Live field.
See RFC 2460 for the definition of the IPv6 Hop Limit field.

5.4.15. protocolIdentifier

Description:
The value of the protocol number in the IP packet header. The protocol number identifies the IP packet payload type. Protocol numbers are defined in the IANA Protocol Numbers registry. In Internet Protocol version 4 (IPv4), this is carried in the Protocol field. In Internet Protocol version 6 (IPv6), this is carried in the Next Header field in the last extension header of the packet.
Abstract Data Type: unsigned8
Data Type Semantics: identifier
ElementId: 4
Status: current
Reference:
See RFC 791 for the specification of the IPv4 protocol field. See RFC 2460 for the specification of the IPv6 protocol field. See the list of protocol numbers assigned by IANA at http://www.iana.org/assignments/protocol-numbers.

5.4.16. nextHeaderIPv6

Description:
The value of the Next Header field of the IPv6 header. The value identifies the type of the following IPv6 extension header or of the following IP payload. Valid values are defined in the IANA Protocol Numbers registry.
Abstract Data Type: unsigned8
ElementId: 193
Status: current
Reference:
See RFC 2460 for the definition of the IPv6 Next Header field. See the list of protocol numbers assigned by IANA at http://www.iana.org/assignments/protocol-numbers.

5.4.17. ipDiffServCodePoint

Description:
The value of a Differentiated Services Code Point (DSCP) encoded in the Differentiated Services field. The Differentiated Services field spans the most significant 6 bits of the IPv4 TOS field or the IPv6 Traffic Class field, respectively.
This Information Element encodes only the 6 bits of the Differentiated Services field. Therefore, its value may range from 0 to 63.
Abstract Data Type: unsigned8
Data Type Semantics: identifier
ElementId: 195
Status: current
Range: The valid range is 0-63.
Reference:
See RFC 3260 for the definition of the Differentiated Services field. See RFC 1812 (Section 5.3.2) and RFC 791 for the definition of the IPv4 TOS field. See RFC 2460 for the definition of the IPv6 Traffic Class field.

5.4.18. ipPrecedence

Description:
The value of the IP Precedence. The IP Precedence value is encoded in the first 3 bits of the IPv4 TOS field or the IPv6 Traffic Class field, respectively. This Information Element encodes only these 3 bits. Therefore, its value may range from 0 to 7.
Abstract Data Type: unsigned8
Data Type Semantics: identifier
ElementId: 196
Status: current

Range: The valid range is 0-7.
Reference:
See RFC 1812 (Section 5.3.3) and RFC 791 for the definition of the IP Precedence. See RFC 1812 (Section 5.3.2) and RFC 791 for the definition of the IPv4 TOS field. See RFC 2460 for the definition of the IPv6 Traffic Class field.

5.4.19. ipClassOfService

Description:
For IPv4 packets, this is the value of the TOS field in the IPv4 packet header. For IPv6 packets, this is the value of the Traffic Class field in the IPv6 packet header.
Abstract Data Type: unsigned8
Data Type Semantics: identifier
ElementId: 5
Status: current
Reference:
See RFC 1812 (Section 5.3.2) and RFC 791 for the definition of the IPv4 TOS field. See RFC 2460 for the definition of the IPv6 Traffic Class field.

5.4.20. postIpClassOfService

Description:
The definition of this Information Element is identical to the definition of Information Element 'ipClassOfService', except that it reports a potentially modified value caused by a middlebox function after the packet passed the Observation Point.
Abstract Data Type: unsigned8
Data Type Semantics: identifier
ElementId: 55
Status: current
Reference:
See RFC 791 for the definition of the IPv4 TOS field. See RFC 2460 for the definition of the IPv6 Traffic Class field. See RFC 3234 for the definition of middleboxes.

5.4.21. flowLabelIPv6

Description:
The value of the IPv6 Flow Label field in the IP packet header.
Abstract Data Type: unsigned32
Data Type Semantics: identifier
ElementId: 31
Status: current
Reference:
See RFC 2460 for the definition of the Flow Label field in the IPv6 packet header.

5.4.22. isMulticast

Description:
If the IP destination address is not a reserved multicast address, then the value of all bits of the octet (including the reserved ones) is zero.
The first bit of this octet is set to 1 if the Version field of the IP header has the value 4 and if the Destination Address field contains a reserved multicast address in the range from 224.0.0.0 to 239.255.255.255. Otherwise, this bit is set to 0. The second and third bits of this octet are reserved for future use.
The remaining bits of the octet are only set to values other than zero if the IP Destination Address is a reserved IPv6 multicast address. Then the fourth bit of the octet is set to the value of the T flag in the IPv6 multicast address and the remaining four bits are set to the value of the scope field in the IPv6 multicast address.

```
   0      1      2      3      4      5      6      7
+------+------+------+------+------+------+------+------+
|   IPv6 multicast scope    |  T   | RES. | RES. | MCv4 |
+------+------+------+------+------+------+------+------+

Bit  0:    set to 1 if IPv4 multicast
Bits 1-2:  reserved for future use
Bit  4:    set to value of T flag, if IPv6 multicast
Bits 4-7:  set to value of multicast scope if IPv6 multicast
```

Abstract Data Type: unsigned8
Data Type Semantics: flags
ElementId: 206
Status: current

Reference:
See RFC 1112 for the specification of reserved IPv4 multicast addresses. See RFC 4291 for the specification of reserved IPv6 multicast addresses and the definition of the T flag and the IPv6 multicast scope.

5.4.23. fragmentIdentification

Description:
The value of the Identification field in the IPv4 packet header or in the IPv6 Fragment header, respectively. The value is 0 for IPv6 if there is no fragment header.
Abstract Data Type: unsigned32
Data Type Semantics: identifier
ElementId: 54
Status: current
Reference:
See RFC 791 for the definition of the IPv4 Identification field. See RFC 2460 for the definition of the Identification field in the IPv6 Fragment header.

5.4.24. fragmentOffset

Description:
The value of the IP fragment offset field in the IPv4 packet header or the IPv6 Fragment header, respectively. The value is 0 for IPv6 if there is no fragment header.
Abstract Data Type: unsigned16
Data Type Semantics: identifier
ElementId: 88
Status: current
Reference:
See RFC 791 for the specification of the fragment offset in the IPv4 header. See RFC 2460 for the specification of the fragment offset in the IPv6 Fragment header.

5.4.25. fragmentFlags

Description:
Fragmentation properties indicated by flags in the IPv4 packet header or the IPv6 Fragment header, respectively.

Bit 0: (RS) Reserved.
The value of this bit MUST be 0 until specified otherwise.

```
   Bit 1:    (DF) 0 = May Fragment,  1 = Don't Fragment.
             Corresponds to the value of the DF flag in the
             IPv4 header.  Will always be 0 for IPv6 unless
             a "don't fragment" feature is introduced to IPv6.

   Bit 2:    (MF) 0 = Last Fragment, 1 = More Fragments.
             Corresponds to the MF flag in the IPv4 header
             or to the M flag in the IPv6 Fragment header,
             respectively.  The value is 0 for IPv6 if there
             is no fragment header.

   Bits 3-7: (DC) Don't Care.
             The values of these bits are irrelevant.

        0   1   2   3   4   5   6   7
      +---+---+---+---+---+---+---+---+
      | R | D | M | D | D | D | D | D |
      | S | F | F | C | C | C | C | C |
      +---+---+---+---+---+---+---+---+
```

Abstract Data Type: unsigned8
Data Type Semantics: flags
ElementId: 197
Status: current
Reference:
See RFC 791 for the specification of the IPv4 fragment flags. See RFC 2460 for the specification of the IPv6 Fragment header.

5.4.26. ipHeaderLength

Description:
The length of the IP header. For IPv6, the value of this Information Element is 40.
Abstract Data Type: unsigned8
ElementId: 189
Status: current
Units: octets
Reference:
See RFC 791 for the specification of the IPv4 header. See RFC 2460 for the specification of the IPv6 header.

5.4.27. ipv4IHL

Description:
The value of the Internet Header Length (IHL) field in the IPv4 header. It specifies the length of the header in units of 4 octets. Please note that its unit is different from most of the other Information Elements reporting length values.

Abstract Data Type: unsigned8
ElementId: 207
Status: current
Units: 4 octets
Reference:
See RFC 791 for the specification of the IPv4 header.

5.4.28. totalLengthIPv4

Description:
The total length of the IPv4 packet.
Abstract Data Type: unsigned16
ElementId: 190
Status: current
Units: octets
Reference:
See RFC 791 for the specification of the IPv4 total length.

5.4.29. ipTotalLength

Description:
The total length of the IP packet.
Abstract Data Type: unsigned64
ElementId: 224
Status: current
Units: octets
Reference:
See RFC 791 for the specification of the IPv4 total length. See RFC 2460 for the specification of the IPv6 payload length. See RFC 2675 for the specification of the IPv6 jumbo payload length.

5.4.30. payloadLengthIPv6

Description:
This Information Element reports the value of the Payload Length field in the IPv6 header. Note that IPv6 extension headers belong to the payload. Also note that in case of a jumbo payload option the value of the Payload Length field in the IPv6 header is zero and so will be the value reported by this Information Element.
Abstract Data Type: unsigned16
ElementId: 191
Status: current
Units: octets
Reference:
See RFC 2460 for the specification of the IPv6 payload length. See RFC 2675 for the specification of the IPv6 jumbo payload option.

5.5. Transport Header Fields

The set of Information Elements related to transport header fields and length includes the Information Elements listed in the table below.

ID	Name	ID	Name
7	sourceTransportPort	238	tcpWindowScale
11	destinationTransportPort	187	tcpUrgentPointer
180	udpSourcePort	188	tcpHeaderLength
181	udpDestinationPort	32	icmpTypeCodeIPv4
205	udpMessageLength	176	icmpTypeIPv4
182	tcpSourcePort	177	icmpCodeIPv4
183	tcpDestinationPort	139	icmpTypeCodeIPv6
184	tcpSequenceNumber	178	icmpTypeIPv6
185	tcpAcknowledgementNumber	179	icmpCodeIPv6
186	tcpWindowSize	33	igmpType

5.5.1. sourceTransportPort

Description:
The source port identifier in the transport header. For the transport protocols UDP, TCP, and SCTP, this is the source port number given in the respective header. This field MAY also be used for future transport protocols that have 16-bit source port identifiers.
Abstract Data Type: unsigned16
Data Type Semantics: identifier
ElementId: 7
Status: current
Reference:
See RFC 768 for the definition of the UDP source port field. See RFC 793 for the definition of the TCP source port field. See RFC 4960 for the definition of SCTP.
Additional information on defined UDP and TCP port numbers can be found at http://www.iana.org/assignments/port-numbers.

5.5.2. destinationTransportPort

Description:
The destination port identifier in the transport header. For the transport protocols UDP, TCP, and SCTP, this is the destination port number given in the respective header. This field MAY also be used for future transport protocols that have 16-bit destination port identifiers.

Abstract Data Type: unsigned16
Data Type Semantics: identifier
ElementId: 11
Status: current
Reference:
See RFC 768 for the definition of the UDP destination port field.
See RFC 793 for the definition of the TCP destination port field.
See RFC 4960 for the definition of SCTP. Additional information on defined UDP and TCP port numbers can be found at http://www.iana.org/assignments/port-numbers.

5.5.3. udpSourcePort

Description:
The source port identifier in the UDP header.
Abstract Data Type: unsigned16
Data Type Semantics: identifier
ElementId: 180
Status: current
Reference:
See RFC 768 for the definition of the UDP source port field.
Additional information on defined UDP port numbers can be found at http://www.iana.org/assignments/port-numbers.

5.5.4. udpDestinationPort

Description:
The destination port identifier in the UDP header.
Abstract Data Type: unsigned16
Data Type Semantics: identifier
ElementId: 181
Status: current
Reference:
See RFC 768 for the definition of the UDP destination port field.
Additional information on defined UDP port numbers can be found at http://www.iana.org/assignments/port-numbers.

5.5.5. udpMessageLength

Description:
The value of the Length field in the UDP header.
Abstract Data Type: unsigned16
ElementId: 205
Status: current
Units: octets
Reference:
See RFC 768 for the specification of the UDP header.

5.5.6. tcpSourcePort

Description:
The source port identifier in the TCP header.
Abstract Data Type: unsigned16
Data Type Semantics: identifier
ElementId: 182
Status: current
Reference:
See RFC 793 for the definition of the TCP source port field.
Additional information on defined TCP port numbers can be found at
http://www.iana.org/assignments/port-numbers.

5.5.7. tcpDestinationPort

Description:
The destination port identifier in the TCP header.
Abstract Data Type: unsigned16
Data Type Semantics: identifier
ElementId: 183
Status: current
Reference:
See RFC 793 for the definition of the TCP destination port field.
Additional information on defined TCP port numbers can be found at
http://www.iana.org/assignments/port-numbers.

5.5.8. tcpSequenceNumber

Description:
The sequence number in the TCP header.
Abstract Data Type: unsigned32
ElementId: 184
Status: current
Reference:
See RFC 793 for the definition of the TCP sequence number.

5.5.9. tcpAcknowledgementNumber

Description:
The acknowledgement number in the TCP header.
Abstract Data Type: unsigned32
ElementId: 185
Status: current
Reference:
See RFC 793 for the definition of the TCP acknowledgement number.

5.5.10. tcpWindowSize

Description:
The window field in the TCP header. If the TCP window scale is supported, then TCP window scale must be known to fully interpret the value of this information.
Abstract Data Type: unsigned16
ElementId: 186
Status: current
Reference:
See RFC 793 for the definition of the TCP window field. See RFC 1323 for the definition of the TCP window scale.

5.5.11. tcpWindowScale

Description:
The scale of the window field in the TCP header.
Abstract Data Type: unsigned16
ElementId: 238
Status: current
Reference:
See RFC 1323 for the definition of the TCP window scale.

5.5.12. tcpUrgentPointer

Description:
The urgent pointer in the TCP header.
Abstract Data Type: unsigned16
ElementId: 187
Status: current
Reference:
See RFC 793 for the definition of the TCP urgent pointer.

5.5.13. tcpHeaderLength

Description:
The length of the TCP header. Note that the value of this Information Element is different from the value of the Data Offset field in the TCP header. The Data Offset field indicates the length of the TCP header in units of 4 octets. This Information Elements specifies the length of the TCP header in units of octets.
Abstract Data Type: unsigned8
ElementId: 188
Status: current
Units: octets
Reference:
See RFC 793 for the definition of the TCP header.

5.5.14. icmpTypeCodeIPv4

Description:
Type and Code of the IPv4 ICMP message. The combination of both values is reported as (ICMP type * 256) + ICMP code.
Abstract Data Type: unsigned16
Data Type Semantics: identifier
ElementId: 32
Status: current
Reference:
See RFC 792 for the definition of the IPv4 ICMP type and code fields.

5.5.15. icmpTypeIPv4

Description:
Type of the IPv4 ICMP message.
Abstract Data Type: unsigned8
Data Type Semantics: identifier
ElementId: 176
Status: current
Reference:
See RFC 792 for the definition of the IPv4 ICMP type field.

5.5.16. icmpCodeIPv4

Description:
Code of the IPv4 ICMP message.
Abstract Data Type: unsigned8
Data Type Semantics: identifier
ElementId: 177
Status: current
Reference:
See RFC 792 for the definition of the IPv4 ICMP code field.

5.5.17. icmpTypeCodeIPv6

Description:
Type and Code of the IPv6 ICMP message. The combination of both values is reported as (ICMP type * 256) + ICMP code.
Abstract Data Type: unsigned16
Data Type Semantics: identifier
ElementId: 139
Status: current
Reference:
See RFC 4443 for the definition of the IPv6 ICMP type and code fields.

5.5.18. icmpTypeIPv6

Description:
Type of the IPv6 ICMP message.
Abstract Data Type: unsigned8
Data Type Semantics: identifier
ElementId: 178
Status: current
Reference:
See RFC 4443 for the definition of the IPv6 ICMP type field.

5.5.19. icmpCodeIPv6

Description:
Code of the IPv6 ICMP message.
Abstract Data Type: unsigned8
Data Type Semantics: identifier
ElementId: 179
Status: current
Reference:
See RFC 4443 for the definition of the IPv6 ICMP code field.

5.5.20. igmpType

Description:
The type field of the IGMP message.
Abstract Data Type: unsigned8
Data Type Semantics: identifier
ElementId: 33
Status: current
Reference:
See RFC 3376 for the definition of the IGMP type field.

5.6. Sub-IP Header Fields

The set of Information Elements related to Sub-IP header fields includes the Information Elements listed in the table below.

ID	Name	ID	Name
56	sourceMacAddress	201	mplsLabelStackLength
81	postSourceMacAddress	194	mplsPayloadLength
58	vlanId	70	mplsTopLabelStackSection
59	postVlanId	71	mplsLabelStackSection2
80	destinationMacAddress	72	mplsLabelStackSection3
57	postDestinationMacAddress	73	mplsLabelStackSection4
146	wlanChannelId	74	mplsLabelStackSection5
147	wlanSSID	75	mplsLabelStackSection6
200	mplsTopLabelTTL	76	mplsLabelStackSection7
203	mplsTopLabelExp	77	mplsLabelStackSection8
237	postMplsTopLabelExp	78	mplsLabelStackSection9
202	mplsLabelStackDepth	79	mplsLabelStackSection10

5.6.1. sourceMacAddress

Description:
The IEEE 802 source MAC address field.
Abstract Data Type: macAddress
Data Type Semantics: identifier
ElementId: 56
Status: current
Reference:
See IEEE.802-3.2002.

5.6.2. postSourceMacAddress

Description:
The definition of this Information Element is identical to the definition of Information Element ’sourceMacAddress’, except that it reports a potentially modified value caused by a middlebox function after the packet passed the Observation Point.
Abstract Data Type: macAddress
Data Type Semantics: identifier
ElementId: 81
Status: current
Reference:
See IEEE.802-3.2002.

5.6.3. vlanId

Description:
The IEEE 802.1Q VLAN identifier (VID) extracted from the Tag Control Information field that was attached to the IP packet.
Abstract Data Type: unsigned16
Data Type Semantics: identifier
ElementId: 58
Status: current
Reference:
See IEEE.802-1Q.2003.

5.6.4. postVlanId

Description:
The definition of this Information Element is identical to the definition of Information Element 'vlanId', except that it reports a potentially modified value caused by a middlebox function after the packet passed the Observation Point.
Abstract Data Type: unsigned16
Data Type Semantics: identifier
ElementId: 59
Status: current
Reference:
See IEEE.802-1Q.2003.

5.6.5. destinationMacAddress

Description:
The IEEE 802 destination MAC address field.
Abstract Data Type: macAddress
Data Type Semantics: identifier
ElementId: 80
Status: current
Reference:
See IEEE.802-3.2002.

5.6.6. postDestinationMacAddress

Description:
The definition of this Information Element is identical to the definition of Information Element 'destinationMacAddress', except that it reports a potentially modified value caused by a middlebox function after the packet passed the Observation Point.
Abstract Data Type: macAddress
Data Type Semantics: identifier
ElementId: 57
Status: current

Reference:
See IEEE.802-3.2002.

5.6.7. wlanChannelId

Description:
The identifier of the 802.11 (Wi-Fi) channel used.
Abstract Data Type: unsigned8
Data Type Semantics: identifier
ElementId: 146
Status: current
Reference:
See IEEE.802-11.1999.

5.6.8. wlanSSID

Description:
The Service Set IDentifier (SSID) identifying an 802.11 (Wi-Fi) network used. According to IEEE.802-11.1999, the SSID is encoded into a string of up to 32 characters.
Abstract Data Type: string
ElementId: 147
Status: current
Reference:
See IEEE.802-11.1999.

5.6.9. mplsTopLabelTTL

Description:
The TTL field from the top MPLS label stack entry, i.e., the last label that was pushed.
Abstract Data Type: unsigned8
ElementId: 200
Status: current
Units: hops
Reference:
See RFC 3032 for the specification of the TTL field.

5.6.10. mplsTopLabelExp

Description:
The Exp field from the top MPLS label stack entry, i.e., the last label that was pushed.

Bits 0-4: Don't Care, value is irrelevant.
Bits 5-7: MPLS Exp field.

```
      0   1   2   3   4   5   6   7
    +---+---+---+---+---+---+---+---+
    |     don't care    |    Exp    |
    +---+---+---+---+---+---+---+---+
```

Abstract Data Type: unsigned8
Data Type Semantics: flags
ElementId: 203
Status: current
Reference:
See RFC 3032 for the specification of the Exp field. See RFC 3270 for usage of the Exp field.

5.6.11. postMplsTopLabelExp

Description:
The definition of this Information Element is identical to the definition of Information Element 'mplsTopLabelExp', except that it reports a potentially modified value caused by a middlebox function after the packet passed the Observation Point.
Abstract Data Type: unsigned8
Data Type Semantics: flags
ElementId: 237
Status: current
Reference:
See RFC 3032 for the specification of the Exp field. See RFC 3270 for usage of the Exp field.

5.6.12. mplsLabelStackDepth

Description:
The number of labels in the MPLS label stack.
Abstract Data Type: unsigned32
ElementId: 202
Status: current
Units: label stack entries
Reference:
See RFC 3032 for the specification of the MPLS label stack.

5.6.13. mplsLabelStackLength

Description:
The length of the MPLS label stack in units of octets.
Abstract Data Type: unsigned32
ElementId: 201
Status: current
Units: octets
Reference:
See RFC 3032 for the specification of the MPLS label stack.

5.6.14. mplsPayloadLength

Description:
The size of the MPLS packet without the label stack.
Abstract Data Type: unsigned32
ElementId: 194
Status: current
Units: octets
Reference:
See RFC 3031 for the specification of MPLS packets. See RFC 3032 for the specification of the MPLS label stack.

5.6.15. mplsTopLabelStackSection

Description:
The Label, Exp, and S fields from the top MPLS label stack entry, i.e., from the last label that was pushed. The size of this Information Element is 3 octets.

```
 0                   1                   2
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Label                  | Exp |S|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Label: Label Value, 20 bits
Exp: Experimental Use, 3 bits
S: Bottom of Stack, 1 bit

Abstract Data Type: octetArray
Data Type Semantics: identifier
ElementId: 70
Status: current
Reference:
See RFC 3032.

5.6.16. mplsLabelStackSection2

Description:
The Label, Exp, and S fields from the label stack entry that was pushed immediately before the label stack entry that would be reported by mplsTopLabelStackSection. See the definition of mplsTopLabelStackSection for further details. The size of this Information Element is 3 octets.

Abstract Data Type: octetArray
Data Type Semantics: identifier
ElementId: 71
Status: current
Reference:
See RFC 3032.

5.6.17. mplsLabelStackSection3

Description:
The Label, Exp, and S fields from the label stack entry that was pushed immediately before the label stack entry that would be reported by mplsLabelStackSection2. See the definition of mplsTopLabelStackSection for further details. The size of this Information Element is 3 octets.

Abstract Data Type: octetArray
Data Type Semantics: identifier
ElementId: 72
Status: current
Reference:
See RFC 3032.

5.6.18. mplsLabelStackSection4

Description:
The Label, Exp, and S fields from the label stack entry that was pushed immediately before the label stack entry that would be reported by mplsLabelStackSection3. See the definition of mplsTopLabelStackSection for further details. The size of this Information Element is 3 octets.

Abstract Data Type: octetArray
Data Type Semantics: identifier
ElementId: 73
Status: current
Reference:
See RFC 3032.

5.6.19. mplsLabelStackSection5

Description:
The Label, Exp, and S fields from the label stack entry that was pushed immediately before the label stack entry that would be reported by mplsLabelStackSection4. See the definition of mplsTopLabelStackSection for further details. The size of this Information Element is 3 octets.
Abstract Data Type: octetArray
Data Type Semantics: identifier
ElementId: 74
Status: current
Reference:
See RFC 3032.

5.6.20. mplsLabelStackSection6

Description:
The Label, Exp, and S fields from the label stack entry that was pushed immediately before the label stack entry that would be reported by mplsLabelStackSection5. See the definition of mplsTopLabelStackSection for further details. The size of this Information Element is 3 octets.
Abstract Data Type: octetArray
Data Type Semantics: identifier
ElementId: 75
Status: current
Reference:
See RFC 3032.

5.6.21. mplsLabelStackSection7

Description:
The Label, Exp, and S fields from the label stack entry that was pushed immediately before the label stack entry that would be reported by mplsLabelStackSection6. See the definition of mplsTopLabelStackSection for further details. The size of this Information Element is 3 octets.
Abstract Data Type: octetArray
Data Type Semantics: identifier
ElementId: 76
Status: current
Reference:
See RFC 3032.

5.6.22. mplsLabelStackSection8

Description:
The Label, Exp, and S fields from the label stack entry that was pushed immediately before the label stack entry that would be reported by mplsLabelStackSection7. See the definition of mplsTopLabelStackSection for further details. The size of this Information Element is 3 octets.
Abstract Data Type: octetArray
Data Type Semantics: identifier
ElementId: 77
Status: current
Reference:
See RFC 3032.

5.6.23. mplsLabelStackSection9

Description:
The Label, Exp, and S fields from the label stack entry that was pushed immediately before the label stack entry that would be reported by mplsLabelStackSection8. See the definition of mplsTopLabelStackSection for further details. The size of this Information Element is 3 octets.
Abstract Data Type: octetArray
Data Type Semantics: identifier
ElementId: 78
Status: current
Reference:
See RFC 3032.

5.6.24. mplsLabelStackSection10

Description:
The Label, Exp, and S fields from the label stack entry that was pushed immediately before the label stack entry that would be reported by mplsLabelStackSection9. See the definition of mplsTopLabelStackSection for further details. The size of this Information Element is 3 octets.
Abstract Data Type: octetArray
Data Type Semantics: identifier
ElementId: 79
Status: current
Reference:
See RFC 3032.

5.7. Derived Packet Properties

The set of Information Elements derived from packet properties (for example, values of header fields) includes the Information Elements listed in the table below.

ID	Name	ID	Name
204	ipPayloadLength	18	bgpNextHopIPv4Address
15	ipNextHopIPv4Address	63	bgpNextHopIPv6Address
62	ipNextHopIPv6Address	46	mplsTopLabelType
16	bgpSourceAsNumber	47	mplsTopLabelIPv4Address
17	bgpDestinationAsNumber	140	mplsTopLabelIPv6Address
128	bgpNextAdjacentAsNumber	90	mplsVpnRouteDistinguisher
129	bgpPrevAdjacentAsNumber		

5.7.1. ipPayloadLength

Description:
The effective length of the IP payload. For IPv4 packets, the value of this Information Element is the difference between the total length of the IPv4 packet (as reported by Information Element totalLengthIPv4) and the length of the IPv4 header (as reported by Information Element headerLengthIPv4). For IPv6, the value of the Payload Length field in the IPv6 header is reported except in the case that the value of this field is zero and that there is a valid jumbo payload option. In this case, the value of the Jumbo Payload Length field in the jumbo payload option is reported.
Abstract Data Type: unsigned32
ElementId: 204
Status: current
Units: octets
Reference:
See RFC 791 for the specification of IPv4 packets. See RFC 2460 for the specification of the IPv6 payload length. See RFC 2675 for the specification of the IPv6 jumbo payload length.

5.7.2. ipNextHopIPv4Address

Description:
The IPv4 address of the next IPv4 hop.
Abstract Data Type: ipv4Address
Data Type Semantics: identifier
ElementId: 15
Status: current

5.7.3. ipNextHopIPv6Address

Description:

The IPv6 address of the next IPv6 hop.
Abstract Data Type: ipv6Address
Data Type Semantics: identifier
ElementId: 62
Status: current

5.7.4. bgpSourceAsNumber

Description:
The autonomous system (AS) number of the source IP address. If AS path information for this Flow is only available as an unordered AS set (and not as an ordered AS sequence), then the value of this Information Element is 0.
Abstract Data Type: unsigned32
Data Type Semantics: identifier
ElementId: 16
Status: current
Reference:
See RFC 4271 for a description of BGP-4, and see RFC 1930 for the definition of the AS number.

5.7.5. bgpDestinationAsNumber

Description:
The autonomous system (AS) number of the destination IP address. If AS path information for this Flow is only available as an unordered AS set (and not as an ordered AS sequence), then the value of this Information Element is 0.
Abstract Data Type: unsigned32
Data Type Semantics: identifier
ElementId: 17
Status: current
Reference:
See RFC 4271 for a description of BGP-4, and see RFC 1930 for the definition of the AS number.

5.7.6. bgpNextAdjacentAsNumber

Description:
The autonomous system (AS) number of the first AS in the AS path to the destination IP address. The path is deduced by looking up the destination IP address of the Flow in the BGP routing information base. If AS path information for this Flow is only available as an unordered AS set (and not as an ordered AS sequence), then the value of this Information Element is 0.
Abstract Data Type: unsigned32
Data Type Semantics: identifier
ElementId: 128

Status: current
Reference:
See RFC 4271 for a description of BGP-4, and see RFC 1930 for the definition of the AS number.

5.7.7. bgpPrevAdjacentAsNumber

Description:
The autonomous system (AS) number of the last AS in the AS path from the source IP address. The path is deduced by looking up the source IP address of the Flow in the BGP routing information base. If AS path information for this Flow is only available as an unordered AS set (and not as an ordered AS sequence), then the value of this Information Element is 0. In case of BGP asymmetry, the bgpPrevAdjacentAsNumber might not be able to report the correct value.
Abstract Data Type: unsigned32
Data Type Semantics: identifier
ElementId: 129
Status: current
Reference:
See RFC 4271 for a description of BGP-4, and see RFC 1930 for the definition of the AS number.

5.7.8. bgpNextHopIPv4Address

Description:
The IPv4 address of the next (adjacent) BGP hop.
Abstract Data Type: ipv4Address
Data Type Semantics: identifier
ElementId: 18
Status: current
Reference:
See RFC 4271 for a description of BGP-4.

5.7.9. bgpNextHopIPv6Address

Description:
The IPv6 address of the next (adjacent) BGP hop.
Abstract Data Type: ipv6Address
Data Type Semantics: identifier
ElementId: 63
Status: current
Reference:
See RFC 4271 for a description of BGP-4.

5.7.10. mplsTopLabelType

Description:
This field identifies the control protocol that allocated the top-of-stack label. Initial values for this field are listed below. Further values may be assigned by IANA in the MPLS label type registry.

- 0x01 TE-MIDPT: Any TE tunnel mid-point or tail label
- 0x02 Pseudowire: Any PWE3 or Cisco AToM based label
- 0x03 VPN: Any label associated with VPN
- 0x04 BGP: Any label associated with BGP or BGP routing
- 0x05 LDP: Any label associated with dynamically assigned labels using LDP

Abstract Data Type: unsigned8
Data Type Semantics: identifier
ElementId: 46
Status: current
Reference:
See RFC 3031 for the MPLS label structure. See RFC 4364 for the association of MPLS labels with Virtual Private Networks (VPNs). See RFC 4271 for BGP and BGP routing. See RFC 5036 for Label Distribution Protocol (LDP). See the list of MPLS label types assigned by IANA at http://www.iana.org/assignments/mpls-label-values.

5.7.11. mplsTopLabelIPv4Address

Description:
The IPv4 address of the system that the MPLS top label will cause this Flow to be forwarded to.
Abstract Data Type: ipv4Address
Data Type Semantics: identifier
ElementId: 47
Status: current
Reference:
See RFC 3031 for the association between MPLS labels and IP addresses.

5.7.12. mplsTopLabelIPv6Address

Description:
The IPv6 address of the system that the MPLS top label will cause this Flow to be forwarded to.
Abstract Data Type: ipv6Address
Data Type Semantics: identifier
ElementId: 140
Status: current
Reference:
See RFC 3031 for the association between MPLS labels and IP addresses.

5.7.13. mplsVpnRouteDistinguisher

Description:
The value of the VPN route distinguisher of a corresponding entry in a VPN routing and forwarding table. Route distinguisher ensures that the same address can be used in several different MPLS VPNs and that it is possible for BGP to carry several completely different routes to that address, one for each VPN. According to RFC 4364, the size of mplsVpnRouteDistinguisher is 8 octets. However, in RFC 4382 an octet string with flexible length was chosen for representing a VPN route distinguisher by object MplsL3VpnRouteDistinguisher. This choice was made in order to be open to future changes of the size. This idea was adopted when choosing octetArray as abstract data type for this Information Element. The maximum length of this Information Element is 256 octets.
Abstract Data Type: octetArray
Data Type Semantics: identifier
ElementId: 90
Status: current
Reference:
See RFC 4364 for the specification of the route distinguisher. See RFC 4382 for the specification of the MPLS/BGP Layer 3 Virtual Private Network (VPN) Management Information Base.

5.8. Min/Max Flow Properties

Information Elements in this section are results of minimum or maximum operations over all packets of a Flow.

ID	Name	ID	Name
25	minimumIpTotalLength	208	ipv4Options
26	maximumIpTotalLength	64	ipv6ExtensionHeaders
52	minimumTTL	6	tcpControlBits
53	maximumTTL	209	tcpOptions

5.8.1. minimumIpTotalLength

Description:
Length of the smallest packet observed for this Flow. The packet length includes the IP header(s) length and the IP payload length.
Abstract Data Type: unsigned64
ElementId: 25
Status: current
Units: octets
Reference:
See RFC 791 for the specification of the IPv4 total length. See RFC 2460 for the specification of the IPv6 payload length. See RFC 2675 for the specification of the IPv6 jumbo payload length.

5.8.2. maximumIpTotalLength

Description:
Length of the largest packet observed for this Flow. The packet length includes the IP header(s) length and the IP payload length.
Abstract Data Type: unsigned64
ElementId: 26
Status: current
Units: octets
Reference:
See RFC 791 for the specification of the IPv4 total length. See RFC 2460 for the specification of the IPv6 payload length. See RFC 2675 for the specification of the IPv6 jumbo payload length.

5.8.3. minimumTTL

Description:
Minimum TTL value observed for any packet in this Flow.
Abstract Data Type: unsigned8
ElementId: 52

Status: current
Units: hops
Reference:
See RFC 791 for the definition of the IPv4 Time to Live field.
See RFC 2460 for the definition of the IPv6 Hop Limit field.

5.8.4. maximumTTL

Description:
Maximum TTL value observed for any packet in this Flow.
Abstract Data Type: unsigned8
ElementId: 53
Status: current
Units: hops
Reference:
See RFC 791 for the definition of the IPv4 Time to Live field.
See RFC 2460 for the definition of the IPv6 Hop Limit field.

5.8.5. ipv4Options

Description:
IPv4 options in packets of this Flow. The information is encoded in a set of bit fields. For each valid IPv4 option type, there is a bit in this set. The bit is set to 1 if any observed packet of this Flow contains the corresponding IPv4 option type. Otherwise, if no observed packet of this Flow contained the respective IPv4 option type, the value of the corresponding bit is 0. The list of valid IPv4 options is maintained by IANA. Note that for identifying an option not just the 5-bit Option Number, but all 8 bits of the Option Type need to match one of the IPv4 options specified at http://www.iana.org/assignments/ip-parameters. Options are mapped to bits according to their option numbers. Option number X is mapped to bit X. The mapping is illustrated by the figure below.

```
        0      1      2      3      4      5      6      7
    +------+------+------+------+------+------+------+------+
    |      | EXP  |   to be assigned by IANA  |  QS  | UMP  | ...
    +------+------+------+------+------+------+------+------+

        8      9     10     11     12     13     14     15
    +------+------+------+------+------+------+------+------+
... | DPS  |NSAPA | SDB  |RTRALT|ADDEXT|  TR  | EIP  |IMITD | ...
    +------+------+------+------+------+------+------+------+
```

```
          16     17     18     19     20     21     22     23
        +------+------+------+------+------+------+------+------+
    ... |ENCODE| VISA | FINN | MTUR | MTUP | ZSU  | SSR  | SID  | ...
        +------+------+------+------+------+------+------+------+

          24     25     26     27     28     29     30     31
        +------+------+------+------+------+------+------+------+
    ... |  RR  |CIPSO |E-SEC |  TS  | LSR  | SEC  | NOP  | EOOL |
        +------+------+------+------+------+------+------+------+
```

```
           Type   Option
       Bit Value  Name    Reference
       ---+-----+-------+------------------------------------
        0     0   EOOL    End of Options List, RFC 791
        1     1   NOP     No Operation, RFC 791
        2   130   SEC     Security, RFC 1108
        3   131   LSR     Loose Source Route, RFC 791
        4    68   TS      Time Stamp, RFC 791
        5   133   E-SEC   Extended Security, RFC 1108
        6   134   CIPSO   Commercial Security
        7     7   RR      Record Route, RFC 791
        8   136   SID     Stream ID, RFC 791
        9   137   SSR     Strict Source Route, RFC 791
       10    10   ZSU     Experimental Measurement
       11    11   MTUP    (obsoleted) MTU Probe, RFC 1191
       12    12   MTUR    (obsoleted) MTU Reply, RFC 1191
       13   205   FINN    Experimental Flow Control
       14   142   VISA    Experimental Access Control
       15    15   ENCODE
       16   144   IMITD   IMI Traffic Descriptor
       17   145   EIP     Extended Internet Protocol, RFC 1385
       18    82   TR      Traceroute, RFC 3193
       19   147   ADDEXT  Address Extension
       20   148   RTRALT  Router Alert, RFC 2113
       21   149   SDB     Selective Directed Broadcast
       22   150   NSAPA   NSAP Address
       23   151   DPS     Dynamic Packet State
       24   152   UMP     Upstream Multicast Pkt.
       25    25   QS      Quick-Start
       30    30   EXP     RFC3692-style Experiment
       30    94   EXP     RFC3692-style Experiment
       30   158   EXP     RFC3692-style Experiment
       30   222   EXP     RFC3692-style Experiment
       ...  ...   ...     Further options numbers
                          may be assigned by IANA
```

Abstract Data Type: unsigned32
Data Type Semantics: flags
ElementId: 208
Status: current
Reference:
See RFC 791 for the definition of IPv4 options. See the list of IPv4 option numbers assigned by IANA at http://www.iana.org/assignments/ip-parameters.

5.8.6. ipv6ExtensionHeaders

Description:
IPv6 extension headers observed in packets of this Flow. The information is encoded in a set of bit fields. For each IPv6 option header, there is a bit in this set. The bit is set to 1 if any observed packet of this Flow contains the corresponding IPv6 extension header. Otherwise, if no observed packet of this Flow contained the respective IPv6 extension header, the value of the corresponding bit is 0.

```
        0     1     2     3     4     5     6     7
    +-----+-----+-----+-----+-----+-----+-----+-----+
    |                  Reserved                     | ...
    +-----+-----+-----+-----+-----+-----+-----+-----+

        8     9    10    11    12    13    14    15
    +-----+-----+-----+-----+-----+-----+-----+-----+
... |                  Reserved                     | ...
    +-----+-----+-----+-----+-----+-----+-----+-----+

       16    17    18    19    20    21    22    23
    +-----+-----+-----+-----+-----+-----+-----+-----+
... |         Reserved            | ESP | AH  | PAY | ...
    +-----+-----+-----+-----+-----+-----+-----+-----+

       24    25    26    27    28    29    30    31
    +-----+-----+-----+-----+-----+-----+-----+-----+
... | DST | HOP | Res | UNK | FRA0| RH  | FRA1| Res |
    +-----+-----+-----+-----+-----+-----+-----+-----+

  Bit    IPv6 Option   Description

 0, Res                Reserved
 1, FRA1     44        Fragmentation header - not first fragment
 2, RH       43        Routing header
 3, FRA0     44        Fragment header - first fragment
 4, UNK                Unknown Layer 4 header
                       (compressed, encrypted, not supported)
```

```
       5, Res                Reserved
       6, HOP       0        Hop-by-hop option header
       7, DST      60        Destination option header
       8, PAY     108        Payload compression header
       9, AH       51        Authentication Header
      10, ESP      50        Encrypted security payload
      11 to 31               Reserved
```

Abstract Data Type: unsigned32
Data Type Semantics: flags
ElementId: 64
Status: current
Reference:

See RFC 2460 for the general definition of IPv6 extension headers and for the specification of the hop-by-hop options header, the routing header, the fragment header, and the destination options header. See RFC 4302 for the specification of the authentication header. See RFC 4303 for the specification of the encapsulating security payload.

5.8.7. tcpControlBits

Description:

TCP control bits observed for packets of this Flow. The information is encoded in a set of bit fields. For each TCP control bit, there is a bit in this set. A bit is set to 1 if any observed packet of this Flow has the corresponding TCP control bit set to 1. A value of 0 for a bit indicates that the corresponding bit was not set in any of the observed packets of this Flow.

```
    0     1     2     3     4     5     6     7
+-----+-----+-----+-----+-----+-----+-----+-----+
|  Reserved | URG | ACK | PSH | RST | SYN | FIN |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

```
Reserved:  Reserved for future use by TCP.  Must be zero.
     URG:  Urgent Pointer field significant
     ACK:  Acknowledgment field significant
     PSH:  Push Function
     RST:  Reset the connection
     SYN:  Synchronize sequence numbers
     FIN:  No more data from sender
```

Abstract Data Type: unsigned8
Data Type Semantics: flags
ElementId: 6
Status: current
Reference:

See RFC 793 for the definition of the TCP control bits in the TCP header.

5.8.8. tcpOptions

Description:

TCP options in packets of this Flow. The information is encoded in a set of bit fields. For each TCP option, there is a bit in this set. The bit is set to 1 if any observed packet of this Flow contains the corresponding TCP option. Otherwise, if no observed packet of this Flow contained the respective TCP option, the value of the corresponding bit is 0.
Options are mapped to bits according to their option numbers. Option number X is mapped to bit X. TCP option numbers are maintained by IANA.

```
        0     1     2     3     4     5     6     7
    +-----+-----+-----+-----+-----+-----+-----+-----+
    |  63 |  62 |  61 |  60 |  59 |  58 |  57 |  56 |  ...
    +-----+-----+-----+-----+-----+-----+-----+-----+

        8     9    10    11    12    13    14    15
    +-----+-----+-----+-----+-----+-----+-----+-----+
... |  55 |  54 |  53 |  52 |  51 |  50 |  49 |  48 |...
    +-----+-----+-----+-----+-----+-----+-----+-----+

       16    17    18    19    20    21    22    23
    +-----+-----+-----+-----+-----+-----+-----+-----+
... |  47 |  46 |  45 |  44 |  43 |  42 |  41 |  40 |...
    +-----+-----+-----+-----+-----+-----+-----+-----+

                          . . .

       56    57    58    59    60    61    62    63
    +-----+-----+-----+-----+-----+-----+-----+-----+
... |   7 |   6 |   5 |   4 |   3 |   2 |   1 |   0 |
    +-----+-----+-----+-----+-----+-----+-----+-----+
```

Abstract Data Type: unsigned64
Data Type Semantics: flags
ElementId: 209
Status: current
Reference:

See RFC 793 for the definition of TCP options. See the list of TCP option numbers assigned by IANA at http://www.iana.org/assignments/tcp-parameters.

5.9. Flow Timestamps

Information Elements in this section are timestamps of events.

Timestamps flowStartSeconds, flowEndSeconds, flowStartMilliseconds, flowEndMilliseconds, flowStartMicroseconds, flowEndMicroseconds, flowStartNanoseconds, flowEndNanoseconds, and systemInitTimeMilliseconds are absolute and have a well-defined fixed time base, such as, for example, the number of seconds since 0000 UTC Jan 1st 1970.

Timestamps flowStartDeltaMicroseconds and flowEndDeltaMicroseconds are relative timestamps only valid within the scope of a single IPFIX Message. They contain the negative time offsets relative to the export time specified in the IPFIX Message Header. The maximum time offset that can be encoded by these delta counters is 1 hour, 11 minutes, and 34.967295 seconds.

Timestamps flowStartSysUpTime and flowEndSysUpTime are relative timestamps indicating the time relative to the last (re-)initialization of the IPFIX Device. For reporting the time of the last (re-)initialization, systemInitTimeMilliseconds can be reported, for example, in Data Records defined by Option Templates.

ID	Name	ID	Name
150	flowStartSeconds	156	flowStartNanoseconds
151	flowEndSeconds	157	flowEndNanoseconds
152	flowStartMilliseconds	158	flowStartDeltaMicroseconds
153	flowEndMilliseconds	159	flowEndDeltaMicroseconds
154	flowStartMicroseconds	160	systemInitTimeMilliseconds
155	flowEndMicroseconds	22	flowStartSysUpTime
		21	flowEndSysUpTime

5.9.1. flowStartSeconds

Description:
The absolute timestamp of the first packet of this Flow.
Abstract Data Type: dateTimeSeconds
ElementId: 150
Status: current
Units: seconds

5.9.2. flowEndSeconds

Description:
The absolute timestamp of the last packet of this Flow.

Abstract Data Type: dateTimeSeconds
ElementId: 151
Status: current
Units: seconds

5.9.3. flowStartMilliseconds

Description:
The absolute timestamp of the first packet of this Flow.
Abstract Data Type: dateTimeMilliseconds
ElementId: 152
Status: current
Units: milliseconds

5.9.4. flowEndMilliseconds

Description:
The absolute timestamp of the last packet of this Flow.
Abstract Data Type: dateTimeMilliseconds
ElementId: 153
Status: current
Units: milliseconds

5.9.5. flowStartMicroseconds

Description:
The absolute timestamp of the first packet of this Flow.
Abstract Data Type: dateTimeMicroseconds
ElementId: 154
Status: current
Units: microseconds

5.9.6. flowEndMicroseconds

Description:
The absolute timestamp of the last packet of this Flow.
Abstract Data Type: dateTimeMicroseconds
ElementId: 155
Status: current
Units: microseconds

5.9.7. flowStartNanoseconds

Description:
The absolute timestamp of the first packet of this Flow.
Abstract Data Type: dateTimeNanoseconds
ElementId: 156
Status: current
Units: nanoseconds

5.9.8. flowEndNanoseconds

Description:
The absolute timestamp of the last packet of this Flow.
Abstract Data Type: dateTimeNanoseconds
ElementId: 157
Status: current
Units: nanoseconds

5.9.9. flowStartDeltaMicroseconds

Description:
This is a relative timestamp only valid within the scope of a single IPFIX Message. It contains the negative time offset of the first observed packet of this Flow relative to the export time specified in the IPFIX Message Header.
Abstract Data Type: unsigned32
ElementId: 158
Status: current
Units: microseconds
Reference:
See the IPFIX protocol specification [RFC5101] for the definition of the IPFIX Message Header.

5.9.10. flowEndDeltaMicroseconds

Description:
This is a relative timestamp only valid within the scope of a single IPFIX Message. It contains the negative time offset of the last observed packet of this Flow relative to the export time specified in the IPFIX Message Header.
Abstract Data Type: unsigned32
ElementId: 159
Status: current
Units: microseconds
Reference:
See the IPFIX protocol specification [RFC5101] for the definition of the IPFIX Message Header.

5.9.11. systemInitTimeMilliseconds

Description:
The absolute timestamp of the last (re-)initialization of the IPFIX Device.
Abstract Data Type: dateTimeMilliseconds
ElementId: 160
Status: current
Units: milliseconds

5.9.12. flowStartSysUpTime

Description:
The relative timestamp of the first packet of this Flow. It indicates the number of milliseconds since the last (re-)initialization of the IPFIX Device (sysUpTime).
Abstract Data Type: unsigned32
ElementId: 22
Status: current
Units: milliseconds

5.9.13. flowEndSysUpTime

Description:
The relative timestamp of the last packet of this Flow. It indicates the number of milliseconds since the last (re-)initialization of the IPFIX Device (sysUpTime).
Abstract Data Type: unsigned32
ElementId: 21
Status: current
Units: milliseconds

5.10. Per-Flow Counters

Information Elements in this section are counters all having integer values. Their values may change for every report they are used in. They cannot serve as part of a Flow Key used for mapping packets to Flows. However, potentially they can be used for selecting exported Flows, for example, by only exporting Flows with more than a threshold number of observed octets.

There are running counters and delta counters. Delta counters are reset to zero each time their values are exported. Running counters continue counting independently of the Exporting Process.

There are per-Flow counters and counters related to the Metering Process and/or the Exporting Process. Per-Flow counters are Flow properties that potentially change each time a packet belonging to the Flow is observed. The set of per-Flow counters includes the Information Elements listed in the table below. Counters related to the Metering Process and/or the Exporting Process are described in Section 5.3.

ID	Name	ID	Name
1	octetDeltaCount	134	droppedOctetTotalCount
23	postOctetDeltaCount	135	droppedPacketTotalCount

198	octetDeltaSumOfSquares	19	postMCastPacketDeltaCount
85	octetTotalCount	20	postMCastOctetDeltaCount
171	postOctetTotalCount	174	postMCastPacketTotalCount
199	octetTotalSumOfSquares	175	postMCastOctetTotalCount
2	packetDeltaCount	218	tcpSynTotalCount
24	postPacketDeltaCount	219	tcpFinTotalCount
86	packetTotalCount	220	tcpRstTotalCount
172	postPacketTotalCount	221	tcpPshTotalCount
132	droppedOctetDeltaCount	222	tcpAckTotalCount
133	droppedPacketDeltaCount	223	tcpUrgTotalCount

5.10.1. octetDeltaCount

Description:
The number of octets since the previous report (if any) in incoming packets for this Flow at the Observation Point. The number of octets includes IP header(s) and IP payload.
Abstract Data Type: unsigned64
Data Type Semantics: deltaCounter
ElementId: 1
Status: current
Units: octets

5.10.2. postOctetDeltaCount

Description:
The definition of this Information Element is identical to the definition of Information Element 'octetDeltaCount', except that it reports a potentially modified value caused by a middlebox function after the packet passed the Observation Point.
Abstract Data Type: unsigned64
Data Type Semantics: deltaCounter
ElementId: 23
Status: current
Units: octets

5.10.3. octetDeltaSumOfSquares

Description:
The sum of the squared numbers of octets per incoming packet since the previous report (if any) for this Flow at the Observation Point. The number of octets includes IP header(s) and IP payload.
Abstract Data Type: unsigned64
ElementId: 198
Status: current

5.10.4. octetTotalCount

Description:
The total number of octets in incoming packets for this Flow at the Observation Point since the Metering Process (re-)initialization for this Observation Point. The number of octets includes IP header(s) and IP payload.
Abstract Data Type: unsigned64
Data Type Semantics: totalCounter
ElementId: 85
Status: current
Units: octets

5.10.5. postOctetTotalCount

Description:
The definition of this Information Element is identical to the definition of Information Element 'octetTotalCount', except that it reports a potentially modified value caused by a middlebox function after the packet passed the Observation Point.
Abstract Data Type: unsigned64
Data Type Semantics: totalCounter
ElementId: 171
Status: current
Units: octets

5.10.6. octetTotalSumOfSquares

Description:
The total sum of the squared numbers of octets in incoming packets for this Flow at the Observation Point since the Metering Process (re-)initialization for this Observation Point. The number of octets includes IP header(s) and IP payload.
Abstract Data Type: unsigned64
ElementId: 199
Status: current
Units: octets

5.10.7. packetDeltaCount

Description:
The number of incoming packets since the previous report (if any) for this Flow at the Observation Point.
Abstract Data Type: unsigned64
Data Type Semantics: deltaCounter
ElementId: 2
Status: current
Units: packets

5.10.8. postPacketDeltaCount

Description:
The definition of this Information Element is identical to the definition of Information Element 'packetDeltaCount', except that it reports a potentially modified value caused by a middlebox function after the packet passed the Observation Point.
Abstract Data Type: unsigned64
Data Type Semantics: deltaCounter
ElementId: 24
Status: current
Units: packets

5.10.9. packetTotalCount

Description:
The total number of incoming packets for this Flow at the Observation Point since the Metering Process (re-)initialization for this Observation Point.
Abstract Data Type: unsigned64
Data Type Semantics: totalCounter
ElementId: 86
Status: current
Units: packets

5.10.10. postPacketTotalCount

Description:
The definition of this Information Element is identical to the definition of Information Element 'packetTotalCount', except that it reports a potentially modified value caused by a middlebox function after the packet passed the Observation Point.
Abstract Data Type: unsigned64
Data Type Semantics: totalCounter
ElementId: 172
Status: current
Units: packets

5.10.11. droppedOctetDeltaCount

Description:
The number of octets since the previous report (if any) in packets of this Flow dropped by packet treatment. The number of octets includes IP header(s) and IP payload.
Abstract Data Type: unsigned64
Data Type Semantics: deltaCounter
ElementId: 132
Status: current
Units: octets

5.10.12. droppedPacketDeltaCount

Description:
The number of packets since the previous report (if any) of this Flow dropped by packet treatment.
Abstract Data Type: unsigned64
Data Type Semantics: deltaCounter
ElementId: 133
Status: current
Units: packets

5.10.13. droppedOctetTotalCount

Description:
The total number of octets in packets of this Flow dropped by packet treatment since the Metering Process (re-)initialization for this Observation Point. The number of octets includes IP header(s) and IP payload.
Abstract Data Type: unsigned64
Data Type Semantics: totalCounter
ElementId: 134
Status: current
Units: octets

5.10.14. droppedPacketTotalCount

Description:
The number of packets of this Flow dropped by packet treatment since the Metering Process (re-)initialization for this Observation Point.
Abstract Data Type: unsigned64
Data Type Semantics: totalCounter
ElementId: 135
Status: current
Units: packets

5.10.15. postMCastPacketDeltaCount

Description:
The number of outgoing multicast packets since the previous report (if any) sent for packets of this Flow by a multicast daemon within the Observation Domain. This property cannot necessarily be observed at the Observation Point, but may be retrieved by other means.
Abstract Data Type: unsigned64
Data Type Semantics: deltaCounter
ElementId: 19
Status: current
Units: packets

5.10.16. postMCastOctetDeltaCount

Description:
The number of octets since the previous report (if any) in outgoing multicast packets sent for packets of this Flow by a multicast daemon within the Observation Domain. This property cannot necessarily be observed at the Observation Point, but may be retrieved by other means. The number of octets includes IP header(s) and IP payload.
Abstract Data Type: unsigned64
Data Type Semantics: deltaCounter
ElementId: 20
Status: current
Units: octets

5.10.17. postMCastPacketTotalCount

Description:
The total number of outgoing multicast packets sent for packets of this Flow by a multicast daemon within the Observation Domain since the Metering Process (re-)initialization. This property cannot necessarily be observed at the Observation Point, but may be retrieved by other means.
Abstract Data Type: unsigned64
Data Type Semantics: totalCounter
ElementId: 174
Status: current
Units: packets

5.10.18. postMCastOctetTotalCount

Description:
The total number of octets in outgoing multicast packets sent for packets of this Flow by a multicast daemon in the Observation Domain since the Metering Process (re-)initialization. This property cannot necessarily be observed at the Observation Point, but may be retrieved by other means. The number of octets includes IP header(s) and IP payload.
Abstract Data Type: unsigned64
Data Type Semantics: totalCounter
ElementId: 175
Status: current
Units: octets

5.10.19. tcpSynTotalCount

Description:
The total number of packets of this Flow with TCP "Synchronize sequence numbers" (SYN) flag set.
Abstract Data Type: unsigned64
Data Type Semantics: totalCounter
ElementId: 218
Status: current
Units: packets
Reference:
See RFC 793 for the definition of the TCP SYN flag.

5.10.20. tcpFinTotalCount

Description:
The total number of packets of this Flow with TCP "No more data from sender" (FIN) flag set.
Abstract Data Type: unsigned64
Data Type Semantics: totalCounter
ElementId: 219
Status: current
Units: packets
Reference:
See RFC 793 for the definition of the TCP FIN flag.

5.10.21. tcpRstTotalCount

Description:
The total number of packets of this Flow with TCP "Reset the connection" (RST) flag set.
Abstract Data Type: unsigned64
Data Type Semantics: totalCounter
ElementId: 220

Status: current
Units: packets
Reference:
See RFC 793 for the definition of the TCP RST flag.

5.10.22. tcpPshTotalCount

Description:
The total number of packets of this Flow with TCP "Push Function" (PSH) flag set.
Abstract Data Type: unsigned64
Data Type Semantics: totalCounter
ElementId: 221
Status: current
Units: packets
Reference:
See RFC 793 for the definition of the TCP PSH flag.

5.10.23. tcpAckTotalCount

Description:
The total number of packets of this Flow with TCP "Acknowledgment field significant" (ACK) flag set.
Abstract Data Type: unsigned64
Data Type Semantics: totalCounter
ElementId: 222
Status: current
Units: packets
Reference:
See RFC 793 for the definition of the TCP ACK flag.

5.10.24. tcpUrgTotalCount

Description:
The total number of packets of this Flow with TCP "Urgent Pointer field significant" (URG) flag set.
Abstract Data Type: unsigned64
Data Type Semantics: totalCounter
ElementId: 223
Status: current
Units: packets
Reference:
See RFC 793 for the definition of the TCP URG flag.

5.11. Miscellaneous Flow Properties

Information Elements in this section describe properties of Flows that are related to Flow start, Flow duration, and Flow termination, but they are not timestamps as the Information Elements in Section 5.9 are.

ID	Name	ID	Name
36	flowActiveTimeout	161	flowDurationMilliseconds
37	flowIdleTimeout	162	flowDurationMicroseconds
136	flowEndReason	61	flowDirection

5.11.1. flowActiveTimeout

Description:
The number of seconds after which an active Flow is timed out anyway, even if there is still a continuous flow of packets.
Abstract Data Type: unsigned16
ElementId: 36
Status: current
Units: seconds

5.11.2. flowIdleTimeout

Description:
A Flow is considered to be timed out if no packets belonging to the Flow have been observed for the number of seconds specified by this field.
Abstract Data Type: unsigned16
ElementId: 37
Status: current
Units: seconds

5.11.3. flowEndReason

Description:
The reason for Flow termination. The range of values includes the following:

0x01: idle timeout
The Flow was terminated because it was considered to be idle.

0x02: active timeout
The Flow was terminated for reporting purposes while it was still active, for example, after the maximum lifetime of unreported Flows was reached.

0x03: end of Flow detected
The Flow was terminated because the Metering Process detected signals indicating the end of the Flow, for example, the TCP FIN flag.

0x04: forced end
The Flow was terminated because of some external event, for example, a shutdown of the Metering Process initiated by a network management application.

      0x05: lack of resources
            The Flow was terminated because of lack of resources
            available to the Metering Process and/or the Exporting
            Process.

   Abstract Data Type: unsigned8
   Data Type Semantics: identifier
   ElementId: 136
   Status: current

5.11.4. flowDurationMilliseconds

   Description:
      The difference in time between the first observed packet of this
      Flow and the last observed packet of this Flow.
   Abstract Data Type: unsigned32
   ElementId: 161
   Status: current
   Units: milliseconds

5.11.5. flowDurationMicroseconds

   Description:
      The difference in time between the first observed packet of this
      Flow and the last observed packet of this Flow.
   Abstract Data Type: unsigned32
   ElementId: 162
   Status: current
   Units: microseconds

5.11.6. flowDirection

   Description:
      The direction of the Flow observed at the Observation Point.
      There are only two values defined.

      0x00: ingress flow
      0x01: egress flow

   Abstract Data Type: unsigned8
   Data Type Semantics: identifier
   ElementId: 61
   Status: current

5.12. Padding

   This section contains a single Information Element that can be used
   for padding of Flow Records.

IPFIX implementations may wish to align Information Elements within Data Records or to align entire Data Records to 4-octet or 8-octet boundaries. This can be achieved by including one or more paddingOctets Information Elements in a Data Record.

ID	Name	ID	Name
210	paddingOctets		

5.12.1. paddingOctets

Description:
The value of this Information Element is always a sequence of 0x00 values.
Abstract Data Type: octetArray
ElementId: 210
Status: current

6. Extending the Information Model

A key requirement for IPFIX is to allow for extending the set of Information Elements that are reported. This section defines the mechanism for extending this set.

Extension can be done by defining new Information Elements. Each new Information Element MUST be assigned a unique Information Element identifier as part of its definition. These unique Information Element identifiers are the connection between the record structure communicated by the protocol using Templates and a consuming application. For generally applicable Information Elements, using IETF and IANA mechanisms to extend the information model is RECOMMENDED.

Names of new Information Elements SHOULD be chosen according to the naming conventions given in Section 2.3.

For extensions, the type space defined in Section 3 can be used. If required, new abstract data types can be added. New abstract data types MUST be defined in IETF Standards Track documents.

Enterprises may wish to define Information Elements without registering them with IANA. IPFIX explicitly supports enterprise-specific Information Elements. Enterprise-specific Information Elements are described in Sections 2.1 and 4.

However, before creating enterprise-specific Information Elements, the general applicability of such Information Elements should be considered. IPFIX does not support enterprise-specific abstract data types.

7. IANA Considerations

7.1. IPFIX Information Elements

This document specifies an initial set of IPFIX Information Elements. The list of these Information Elements with their identifiers is given in Section 4. The Internet Assigned Numbers Authority (IANA) has created a new registry for IPFIX Information Element identifiers and filled it with the initial list in Section 4.

New assignments for IPFIX Information Elements will be administered by IANA through Expert Review [RFC5226], i.e., review by one of a group of experts designated by an IETF Area Director. The group of experts MUST check the requested Information Element for completeness and accuracy of the description and for correct naming according to the naming conventions in Section 2.3. Requests for Information Elements that duplicate the functionality of existing Information Elements SHOULD be declined. The smallest available identifier SHOULD be assigned to a new Information Element.

The specification of new IPFIX Information Elements MUST use the template specified in Section 2.1 and MUST be published using a well-established and persistent publication medium. The experts will initially be drawn from the Working Group Chairs and document editors of the IPFIX and PSAMP Working Groups.

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 registry for MPLS label types and filled it with the initial list from the description Information Element #46, mplsTopLabelType.

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

7.3. XML Namespace and Schema

Appendix B defines an XML schema for IPFIX Information Element definitions. All Information Elements specified in this document are defined by the XML specification in Appendix A that is a valid XML record according to the schema in Appendix B. This schema may also be used for specifying further Information Elements in future extensions of the IPFIX information model in a machine-readable way.

Appendix B uses URNs to describe an XML namespace and an XML schema for IPFIX Information Elements conforming to a registry mechanism described in [RFC3688]. Two URI assignments have been made.

1. Registration for the IPFIX information model namespace
   * URI: urn:ietf:params:xml:ns:ipfix-info
   * Registrant Contact: IETF IPFIX Working Group <ipfix@ietf.org>, as designated by the IESG <iesg@ietf.org>.
   * XML: None. Namespace URIs do not represent an XML.

2. Registration for the IPFIX information model schema
   * URI: urn:ietf:params:xml:schema:ipfix-info
   * Registrant Contact: IETF IPFIX Working Group <ipfix@ietf.org>, as designated by the IESG <iesg@ietf.org>.
   * XML: See Appendix B of this document.

8. Security Considerations

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

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

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

This document does not specify any Information Element carrying keying material. If future extensions will do so, then appropriate precautions need to be taken for properly protecting such sensitive information.

9. Acknowledgements

The editors thank Paul Callato for creating the initial version of this document, and Thomas Dietz for developing the XSLT scripts that generate large portions of the text part of this document from the XML appendices.

10. References

10.1. Normative References

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

[RFC5905] Mills, D., Delaware, U., Martin, J., Burbank, J. and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010

[RFC5101bis] Claise, B., "Specification of the IP Flow Information eXport (IPFIX) Protocol for the Exchange of IP Traffic Flow Information", draft-claise-ipfix-protocol-rfc5101bis-02, Work in Progress, October 2011.

10.2. Informative References

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

[IEEE.802-11.1999] "Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications", IEEE Standard 802.11, 1999, <http://standards.ieee.org/getieee802/download/802.11-1999.pdF>.

[IEEE.802-1Q.2003] Institute of Electrical and Electronics Engineers, "Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks", IEEE Standard 802.1Q, March 2003.

[IEEE.802-3.2002] "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.

[ISO.10646-1.1993]
International Organization for Standardization, "Information Technology - Universal Multiple-octet coded Character Set (UCS) - Part 1: Architecture and Basic Multilingual Plane", ISO Standard 10646-1, May 1993.

[ISO.646.1991]
International Organization for Standardization, "Information technology - ISO 7-bit coded character set for information interchange", ISO Standard 646, 1991.

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, September 1981.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC1108] Kent, S., "U.S. Department of Defense Security Options for the Internet Protocol", RFC 1108, November 1991.

[RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, August 1989.

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, November 1990.

[RFC1323] Jacobson, V., Braden, R., and D. Borman, "TCP Extensions for High Performance", RFC 1323, May 1992.

[RFC1385] Wang, Z., "EIP: The Extended Internet Protocol", RFC 1385, November 1992.

[RFC1812] Baker, F., Ed., "Requirements for IP Version 4 Routers", RFC 1812, June 1995.

[RFC1930] Hawkinson, J. and T. Bates, "Guidelines for creation, selection, and registration of an Autonomous System (AS)", BCP 6, RFC 1930, March 1996.

[RFC2113] Katz, D., "IP Router Alert Option", RFC 2113, February 1997.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[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.

[RFC2675] Borman, D., Deering, S., and R. Hinden, "IPv6 Jumbograms", RFC 2675, August 1999.

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

[RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, January 2001.

[RFC3032] Rosen, E., Tappan, D., Fedorkow, G., Rekhter, Y., Farinacci, D., Li, T., and A. Conta, "MPLS Label Stack Encoding", RFC 3032, January 2001.

[RFC3193] Patel, B., Aboba, B., Dixon, W., Zorn, G., and S. Booth, "Securing L2TP using IPsec", RFC 3193, November 2001.

[RFC3234] Carpenter, B. and S. Brim, "Middleboxes: Taxonomy and Issues", RFC 3234, February 2002.

[RFC3260] Grossman, D., "New Terminology and Clarifications for Diffserv", RFC 3260, April 2002.

[RFC3270] Le Faucheur, F., Wu, L., Davie, B., Davari, S., Vaananen, P., Krishnan, R., Cheval, P., and J. Heinanen, "Multi-Protocol Label Switching (MPLS) Support of Differentiated Services", RFC 3270, May 2002.

[RFC3376] Cain, B., Deering, S., Kouvelas, I., Fenner, B., and A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 3376, October 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.

[RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.

[RFC4302] Kent, S., "IP Authentication Header", RFC 4302, December 2005.

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

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, February 2006.

[RFC4382] Nadeau, T., Ed., and H. van der Linde, Ed., "MPLS/BGP Layer 3 Virtual Private Network (VPN) Management Information Base", RFC 4382, February 2006.

[RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, March 2006.

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

[RFC5036] Andersson, L., Ed., Minei, I., Ed., and B. Thomas, Ed., "LDP Specification", RFC 5036, October 2007.

[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.

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

[RFC5471] Schmoll, C., Aitken, P., and B. Claise, "Guidelines for IP Flow Information Export (IPFIX) Testing", RFC5471, March 2009.

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

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

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

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

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

[IPFIX-CONF] Muenz, G., Claise, B., and P. Aitken, "Configuration Data Model for IPFIX and PSAMP", draft-ietf-ipfix-configuration-model-10, Work in Progress, July 2011.

[IPFIX-MED-PROTO] Claise, B., Kobayashi, A., and B. Trammell, "Specification of the Protocol for IPFIX Mediations", draft-claise-ipfix-mediation-protocol-04, Work in Progress, July 2011.

[RFC5815bis] Dietz, T., Kobayashi, A., Claise, B., and G. Muenz, "Definitions of Managed Objects for IP Flow Information Export", draft-dkcm-ipfix-rfc5815bis-00.txt, Work in Progress, October 2011.

Appendix A. XML Specification of IPFIX Information Elements

This appendix contains a machine-readable description of the IPFIX information model coded in XML. Note that this appendix is of informational nature, while the text in Section 4 (generated from this appendix) is normative.

Using a machine-readable syntax for the information model enables the creation of IPFIX-aware tools that can automatically adapt to extensions to the information model, by simply reading updated information model specifications.

The wide availability of XML-aware tools and libraries for client devices is a primary consideration for this choice. In particular, libraries for parsing XML documents are readily available. Also, mechanisms such as the Extensible Stylesheet Language (XSL) allow for transforming a source XML document into other documents. This document was authored in XML and transformed according to [RFC2629].

It should be noted that the use of XML in Exporters, Collectors, or other tools is not mandatory for the deployment of IPFIX. In particular, Exporting Processes do not produce or consume XML as part of their operation. It is expected that IPFIX Collectors MAY take advantage of the machine readability of the information model vs. hard coding their behavior or inventing proprietary means for accommodating extensions.


```xml
<?xml version="1.0" encoding="UTF-8"?>

<fieldDefinitions xmlns="urn:ietf:params:xml:ns:ipfix-info"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="urn:ietf:params:xml:ns:ipfix-info
             ipfix-info.xsd">

  <field name="lineCardId" dataType="unsigned32"
         group="scope"
         dataTypeSemantics="identifier"
         elementId="141" applicability="option" status="current">
    <description>
      <paragraph>
        An identifier of a line card that is unique per IPFIX
        Device hosting an Observation Point.  Typically, this
        Information Element is used for limiting the scope
        of other Information Elements.
      </paragraph>
    </description>
  </field>
  <field name="portId" dataType="unsigned32"
```

```
              group="scope"
              dataTypeSemantics="identifier"
              elementId="142" applicability="option" status="current">
         <description>
           <paragraph>
             An identifier of a line port that is unique per IPFIX
             Device hosting an Observation Point.  Typically, this
             Information Element is used for limiting the scope
             of other Information Elements.
           </paragraph>
         </description>
       </field>

       <field name="ingressInterface" dataType="unsigned32"
              group="scope"
              dataTypeSemantics="identifier"
              elementId="10" applicability="all" status="current">
         <description>
           <paragraph>
             The index of the IP interface where packets of this Flow
             are being received.  The value matches the value of managed
             object ’ifIndex’ as defined in RFC 2863.
             Note that ifIndex values are not assigned statically to an
             interface and that the interfaces may be renumbered every
             time the device’s management system is re-initialized, as
             specified in RFC 2863.
           </paragraph>
         </description>
         <reference>
           <paragraph>
           See RFC 2863 for the definition of the ifIndex object.
           </paragraph>
         </reference>
       </field>

       <field name="egressInterface" dataType="unsigned32"
              group="scope"
              dataTypeSemantics="identifier"
              elementId="14" applicability="all" status="current">
         <description>
           <paragraph>
             The index of the IP interface where packets of
             this Flow are being sent.  The value matches the value of
             managed object ’ifIndex’ as defined in RFC 2863.
             Note that ifIndex values are not assigned statically to an
             interface and that the interfaces may be renumbered every
             time the device’s management system is re-initialized, as
             specified in RFC 2863.
```

```
         </paragraph>
       </description>
       <reference>
         <paragraph>
         See RFC 2863 for the definition of the ifIndex object.
         </paragraph>
       </reference>
     </field>

     <field name="meteringProcessId" dataType="unsigned32"
            group="scope"
            dataTypeSemantics="identifier"
            elementId="143" applicability="option" status="current">
       <description>
         <paragraph>
           An identifier of a Metering 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 are typically assigned
           dynamically.
           The Metering Process may be re-started with a different ID.
         </paragraph>
       </description>
     </field>

     <field name="exportingProcessId" dataType="unsigned32"
            group="scope"
            dataTypeSemantics="identifier"
            elementId="144" applicability="option" status="current">
       <description>
         <paragraph>
           An identifier of an Exporting 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 are typically assigned
           dynamically.  The Exporting Process may be re-started
           with a different ID.
         </paragraph>
       </description>
     </field>

     <field name="flowId" dataType="unsigned64"
            group="scope"
            dataTypeSemantics="identifier"
            elementId="148" applicability="option" status="current">
       <description>
         <paragraph>
           An identifier of a Flow that is unique within an Observation
```

```
           Domain.  This Information Element can be used to distinguish
           between different Flows if Flow Keys such as IP addresses and
           port numbers are not reported or are reported in separate
           records.
         </paragraph>
       </description>
     </field>

     <field name="templateId" dataType="unsigned16"
            group="scope"
            dataTypeSemantics="identifier"
            elementId="145" applicability="option" status="current">
       <description>
         <paragraph>
           An identifier of a Template that is locally unique within a
           combination of a Transport session and an Observation Domain.
         </paragraph>
         <paragraph>
           Template IDs 0-255 are reserved for Template Sets, Options
           Template Sets, and other reserved Sets yet to be created.
           Template IDs of Data Sets are numbered from 256 to 65535.
         </paragraph>
         <paragraph>
           Typically, this Information Element is used for limiting
           the scope of other Information Elements.
           Note that after a re-start of the Exporting Process Template
           identifiers may be re-assigned.
         </paragraph>
       </description>
     </field>

     <field name="observationDomainId" dataType="unsigned32"
            group="scope"
            dataTypeSemantics="identifier"
            elementId="149" applicability="option" status="current">
       <description>
         <paragraph>
           An identifier of an Observation Domain that is locally
           unique to an Exporting Process.  The Exporting Process uses
           the Observation Domain ID to uniquely identify to the
           Collecting Process the Observation Domain where Flows
           were metered.  It is RECOMMENDED that this identifier is
           also unique per IPFIX Device.
         </paragraph>
         <paragraph>
           A value of 0 indicates that no specific Observation Domain
           is identified by this Information Element.
         </paragraph>
```

```
         <paragraph>
           Typically, this Information Element is used for limiting
           the scope of other Information Elements.
         </paragraph>
       </description>
     </field>

     <field name="observationPointId" dataType="unsigned32"
            group="scope"
            dataTypeSemantics="identifier"
            elementId="138" applicability="option" status="current">
       <description>
         <paragraph>
         An identifier of an Observation Point that is unique per
         Observation Domain.  It is RECOMMENDED that this identifier is
         also unique per IPFIX Device.  Typically, this Information
         Element is used for limiting the scope of other Information
         Elements.
         </paragraph>
       </description>
     </field>
     <field name="commonPropertiesId" dataType="unsigned64"
            group="scope"
            dataTypeSemantics="identifier"
            elementId="137" applicability="option" status="current">
       <description>
         <paragraph>
           An identifier of a set of common properties that is
           unique per Observation Domain and Transport Session.
           Typically, this Information Element is used to link to
           information reported in separate Data Records.
         </paragraph>
       </description>
     </field>

     <field name="exporterIPv4Address" dataType="ipv4Address"
            dataTypeSemantics="identifier"
            group="config"
            elementId="130" applicability="all" status="current">
       <description>
         <paragraph>
         The IPv4 address used by the Exporting Process.  This is used
         by the Collector to identify the Exporter in cases where the
         identity of the Exporter may have been obscured by the use of
         a proxy.
         </paragraph>
       </description>
     </field>
```

```
     <field name="exporterIPv6Address" dataType="ipv6Address"
            dataTypeSemantics="identifier"
            group="config"
            elementId="131" applicability="all" status="current">
       <description>
         <paragraph>
         The IPv6 address used by the Exporting Process.  This is used
         by the Collector to identify the Exporter in cases where the
         identity of the Exporter may have been obscured by the use of
         a proxy.
         </paragraph>
       </description>
     </field>

     <field name="exporterTransportPort" dataType="unsigned16"
            group="config"
            dataTypeSemantics="identifier"
            elementId="217" applicability="all" status="current">
       <description>
         <paragraph>
         The source port identifier from which the Exporting
         Process sends Flow information.  For the transport protocols
         UDP, TCP, and SCTP, this is the source port number.
         This field MAY also be used for future transport protocols
         that have 16-bit source port identifiers.  This field may
         be useful for distinguishing multiple Exporting Processes
         that use the same IP address.
         </paragraph>
       </description>
       <reference>
         <paragraph>
         See RFC 768 for the definition of the UDP
         source port field.
         See RFC 793 for the definition of the TCP
         source port field.
         See RFC 4960 for the definition of SCTP.
         </paragraph>
         <paragraph>
         Additional information on defined UDP and TCP port numbers can
         be found at http://www.iana.org/assignments/port-numbers.
         </paragraph>
       </reference>
     </field>

     <field name="collectorIPv4Address" dataType="ipv4Address"
            dataTypeSemantics="identifier"
            group="config"
            elementId="211" applicability="all" status="current">
```

```
      <description>
        <paragraph>
        An IPv4 address to which the Exporting Process sends Flow
        information.
        </paragraph>
      </description>
    </field>

    <field name="collectorIPv6Address" dataType="ipv6Address"
           dataTypeSemantics="identifier"
           group="config"
           elementId="212" applicability="all" status="current">
      <description>
        <paragraph>
        An IPv6 address to which the Exporting Process sends Flow
        information.
        </paragraph>
      </description>
    </field>

    <field name="exportInterface" dataType="unsigned32"
           group="config"
           dataTypeSemantics="identifier"
           elementId="213" applicability="all" status="current">
      <description>
        <paragraph>
          The index of the interface from which IPFIX Messages sent
          by the Exporting Process to a Collector leave the IPFIX
          Device.  The value matches the value of
          managed object 'ifIndex' as defined in RFC 2863.
          Note that ifIndex values are not assigned statically to an
          interface and that the interfaces may be renumbered every
          time the device's management system is re-initialized, as
          specified in RFC 2863.
        </paragraph>
      </description>
      <reference>
        <paragraph>
        See RFC 2863 for the definition of the ifIndex object.
        </paragraph>
      </reference>
    </field>

    <field name="exportProtocolVersion" dataType="unsigned8"
           dataTypeSemantics="identifier"
           group="config"
           elementId="214" applicability="all" status="current">
      <description>
```

```
         <paragraph>
           The protocol version used by the Exporting Process for
           sending Flow information.  The protocol version is given
           by the value of the Version Number field in the Message
           Header.
         </paragraph>
         <paragraph>
           The protocol version is 10 for IPFIX and 9 for NetFlow
           version 9.
           A value of 0 indicates that no export protocol is in use.
         </paragraph>
       </description>
       <reference>
         <paragraph>
         See the IPFIX protocol specification [RFC5101] for the
         definition of the IPFIX Message Header.
         </paragraph>
         <paragraph>
         See RFC 3954 for the definition of the NetFlow
         version 9 message header.
         </paragraph>
       </reference>
     </field>

     <field name="exportTransportProtocol" dataType="unsigned8"
            group="config"
            dataTypeSemantics="identifier"
            elementId="215" applicability="all" status="current">
       <description>
         <paragraph>
         The value of the protocol number used by the Exporting Process
         for sending Flow information.
         The protocol number identifies the IP packet payload type.
         Protocol numbers are defined in the IANA Protocol Numbers
         registry.
         </paragraph>

         <paragraph>
         In Internet Protocol version 4 (IPv4), this is carried in the
         Protocol field.  In Internet Protocol version 6 (IPv6), this
         is carried in the Next Header field in the last extension
         header of the packet.
         </paragraph>
       </description>
       <reference>
         <paragraph>
         See RFC 791 for the specification of the IPv4
         protocol field.
```

```
         See RFC 2460 for the specification of the IPv6
         protocol field.
         See the list of protocol numbers assigned by IANA at
         http://www.iana.org/assignments/protocol-numbers.
         </paragraph>
       </reference>
     </field>

     <field name="collectorTransportPort" dataType="unsigned16"
            group="config"
            dataTypeSemantics="identifier"
            elementId="216" applicability="all" status="current">
       <description>
         <paragraph>
         The destination port identifier to which the Exporting
         Process sends Flow information.  For the transport protocols
         UDP, TCP, and SCTP, this is the destination port number.
         This field MAY also be used for future transport protocols
         that have 16-bit source port identifiers.
         </paragraph>
       </description>
       <reference>
         <paragraph>
         See RFC 768 for the definition of the UDP
         destination port field.
         See RFC 793 for the definition of the TCP
         destination port field.
         See RFC 4960 for the definition of SCTP.
         </paragraph>
         <paragraph>
         Additional information on defined UDP and TCP port numbers can
         be found at http://www.iana.org/assignments/port-numbers.
         </paragraph>
       </reference>
     </field>

     <field name="flowKeyIndicator" dataType="unsigned64"
            dataTypeSemantics="flags"
            group="config"
            elementId="173" applicability="all" status="current">
       <description>
         <paragraph>
         This set of bit fields is used for marking the Information
         Elements of a Data Record that serve as Flow Key.  Each bit
         represents an Information Element in the Data Record with
         the n-th bit representing the n-th Information Element.
         A bit set to value 1 indicates that the corresponding
         Information Element is a Flow Key of the reported Flow.
```

```
         A bit set to value 0 indicates that this is not the case.
         </paragraph>
         <paragraph>
         If the Data Record contains more than 64 Information Elements,
         the corresponding Template SHOULD be designed such that all
         Flow Keys are among the first 64 Information Elements, because
         the flowKeyIndicator only contains 64 bits.  If the Data Record
         contains less than 64 Information Elements, then the bits in
         the flowKeyIndicator for which no corresponding Information
         Element exists MUST have the value 0.
         </paragraph>
       </description>
     </field>

     <field name="exportedMessageTotalCount" dataType="unsigned64"
            dataTypeSemantics="totalCounter"
            group="processCounter"
            elementId="41" applicability="data" status="current">
       <description>
         <paragraph>
           The total number of IPFIX Messages that the Exporting Process
           has sent since the Exporting Process (re-)initialization to
           a particular Collecting Process.
           The reported number excludes the IPFIX Message that carries
           the counter value.
           If this Information Element is sent to a particular
           Collecting Process, then by default it specifies the number
           of IPFIX Messages sent to this Collecting Process.
         </paragraph>
       </description>
       <units>messages</units>
     </field>

     <field name="exportedOctetTotalCount" dataType="unsigned64"
            dataTypeSemantics="totalCounter"
            group="processCounter"
            elementId="40" applicability="data" status="current">
       <description>
         <paragraph>
           The total number of octets that the Exporting Process
           has sent since the Exporting Process (re-)initialization
           to a particular Collecting Process.
           The value of this Information Element is calculated by
           summing up the IPFIX Message Header length values of all
           IPFIX Messages that were successfully sent to the Collecting
           Process.  The reported number excludes octets in the IPFIX
           Message that carries the counter value.
           If this Information Element is sent to a particular
```

```
           Collecting Process, then by default it specifies the number
           of octets sent to this Collecting Process.
         </paragraph>
       </description>
       <units>octets</units>
     </field>

     <field name="exportedFlowRecordTotalCount" dataType="unsigned64"
            group="processCounter"
            dataTypeSemantics="totalCounter"
            elementId="42" applicability="data" status="current">
       <description>
         <paragraph>
           The total number of Flow Records that the Exporting
           Process has sent as Data Records since the Exporting
           Process (re-)initialization to a particular Collecting
           Process.  The reported number excludes Flow Records in
           the IPFIX Message that carries the counter value.
           If this Information Element is sent to a particular
           Collecting Process, then by default it specifies the number
           of Flow Records sent to this process.
         </paragraph>
       </description>
       <units>flows</units>
     </field>

     <field name="observedFlowTotalCount" dataType="unsigned64"
            dataTypeSemantics="totalCounter"
            group="processCounter"
            elementId="163" applicability="data" status="current">
       <description>
         <paragraph>
         The total number of Flows observed in the Observation Domain
         since the Metering Process (re-)initialization for this
         Observation Point.
         </paragraph>
       </description>
       <units>flows</units>
     </field>

     <field name="ignoredPacketTotalCount" dataType="unsigned64"
            dataTypeSemantics="totalCounter"
            group="processCounter"
            elementId="164" applicability="data" status="current">
       <description>
         <paragraph>
            The total number of observed IP packets that the
            Metering Process did not process since the
```

```
            (re-)initialization of the Metering Process.
         </paragraph>
       </description>
       <units>packets</units>
     </field>

     <field name="ignoredOctetTotalCount" dataType="unsigned64"
            dataTypeSemantics="totalCounter"
            group="processCounter"
            elementId="165" applicability="data" status="current">
       <description>
         <paragraph>
            The total number of octets in observed IP packets
            (including the IP header) that the Metering Process
            did not process since the (re-)initialization of the
            Metering Process.
         </paragraph>
       </description>
       <units>octets</units>
     </field>

     <field name="notSentFlowTotalCount" dataType="unsigned64"
            dataTypeSemantics="totalCounter"
            group="processCounter"
            elementId="166" applicability="data" status="current">
       <description>
         <paragraph>
            The total number of Flow Records that were generated by the
            Metering Process and dropped by the Metering Process or
            by the Exporting Process instead of being sent to the
            Collecting Process.  There are several potential reasons for
            this including resource shortage and special Flow export
            policies.
         </paragraph>
       </description>
       <units>flows</units>
     </field>

     <field name="notSentPacketTotalCount" dataType="unsigned64"
            dataTypeSemantics="totalCounter"
            group="processCounter"
            elementId="167" applicability="data" status="current">
       <description>
         <paragraph>
            The total number of packets in Flow Records that were
            generated by the Metering Process and dropped
            by the Metering Process or by the Exporting Process
            instead of being sent to the Collecting Process.
```

```
            There are several potential reasons for this including
            resource shortage and special Flow export policies.
          </paragraph>
        </description>
        <units>packets</units>
      </field>

      <field name="notSentOctetTotalCount" dataType="unsigned64"
             dataTypeSemantics="totalCounter"
             group="processCounter"
             elementId="168" applicability="data" status="current">
        <description>
          <paragraph>
            The total number of octets in packets in Flow Records
            that were generated by the Metering Process and
            dropped by the Metering Process or by the Exporting
            Process instead of being sent to the Collecting Process.
            There are several potential reasons for this including
            resource shortage and special Flow export policies.
          </paragraph>
        </description>
        <units>octets</units>
      </field>

      <field name="ipVersion" dataType="unsigned8"
             group="ipHeader"
             dataTypeSemantics="identifier"
             elementId="60" applicability="all" status="current">
        <description>
          <paragraph>
          The IP version field in the IP packet header.
          </paragraph>
        </description>
        <reference>
          <paragraph>
          See RFC 791 for the definition of the version field
          in the IPv4 packet header.
          See RFC 2460 for the definition of the version field
          in the IPv6 packet header.
          Additional information on defined version numbers
          can be found at
          http://www.iana.org/assignments/version-numbers.
          </paragraph>
        </reference>
      </field>

      <field name="sourceIPv4Address" dataType="ipv4Address"
             group="ipHeader"
```

```
            dataTypeSemantics="identifier"
            elementId="8" applicability="all" status="current">
       <description>
         <paragraph>
         The IPv4 source address in the IP packet header.
         </paragraph>
       </description>
       <reference>
         <paragraph>
         See RFC 791 for the definition of the IPv4 source
         address field.
         </paragraph>
       </reference>
     </field>

     <field name="sourceIPv6Address" dataType="ipv6Address"
            group="ipHeader"
            dataTypeSemantics="identifier"
            elementId="27" applicability="all" status="current">
       <description>
         <paragraph>
         The IPv6 source address in the IP packet header.
         </paragraph>
       </description>
       <reference>
         <paragraph>
         See RFC 2460 for the definition of the
         Source Address field in the IPv6 header.
         </paragraph>
       </reference>
     </field>

     <field name="sourceIPv4PrefixLength" dataType="unsigned8"
            group="ipHeader"
            elementId="9" applicability="option" status="current">
       <description>
         <paragraph>
         The number of contiguous bits that are relevant in the
         sourceIPv4Prefix Information Element.
         </paragraph>
       </description>
       <units>bits</units>
       <range>0-32</range>
     </field>

     <field name="sourceIPv6PrefixLength" dataType="unsigned8"
            group="ipHeader"
            elementId="29" applicability="option" status="current">
```

```
        <description>
          <paragraph>
          The number of contiguous bits that are relevant in the
          sourceIPv6Prefix Information Element.
          </paragraph>
        </description>
        <units>bits</units>
        <range>0-128</range>
      </field>

      <field name="sourceIPv4Prefix" dataType="ipv4Address"
             group="ipHeader"
             elementId="44" applicability="data" status="current">
        <description>
          <paragraph>
          IPv4 source address prefix.
          </paragraph>
        </description>
      </field>

      <field name="sourceIPv6Prefix" dataType="ipv6Address"
             group="ipHeader"
             elementId="170" applicability="data" status="current">
        <description>
          <paragraph>
          IPv6 source address prefix.
          </paragraph>
        </description>
      </field>

      <field name="destinationIPv4Address" dataType="ipv4Address"
             group="ipHeader"
             dataTypeSemantics="identifier"
             elementId="12" applicability="all" status="current">
        <description>
          <paragraph>
          The IPv4 destination address in the IP packet header.
          </paragraph>
        </description>
        <reference>
          <paragraph>
          See RFC 791 for the definition of the IPv4
          destination address field.
          </paragraph>
        </reference>
      </field>

      <field name="destinationIPv6Address" dataType="ipv6Address"
```

```
            group="ipHeader"
            dataTypeSemantics="identifier"
            elementId="28" applicability="all" status="current">
       <description>
         <paragraph>
         The IPv6 destination address in the IP packet header.
         </paragraph>
       </description>
       <reference>
         <paragraph>
         See RFC 2460 for the definition of the
         Destination Address field in the IPv6 header.
         </paragraph>
       </reference>
     </field>

     <field name="destinationIPv4PrefixLength" dataType="unsigned8"
            group="ipHeader"
            elementId="13" applicability="option" status="current">
       <description>
         <paragraph>
         The number of contiguous bits that are relevant in the
         destinationIPv4Prefix Information Element.
         </paragraph>
       </description>
       <units>bits</units>
       <range>0-32</range>
     </field>

     <field name="destinationIPv6PrefixLength" dataType="unsigned8"
            group="ipHeader"
            elementId="30" applicability="option" status="current">
       <description>
         <paragraph>
         The number of contiguous bits that are relevant in the
         destinationIPv6Prefix Information Element.
         </paragraph>
       </description>
       <units>bits</units>
       <range>0-128</range>
     </field>

     <field name="destinationIPv4Prefix" dataType="ipv4Address"
            group="ipHeader"
            elementId="45" applicability="data" status="current">
       <description>
         <paragraph> IPv4 destination address prefix. </paragraph>
       </description>
```

```
</field>

<field name="destinationIPv6Prefix" dataType="ipv6Address"
       group="ipHeader"
       elementId="169" applicability="data" status="current">
  <description>
    <paragraph> IPv6 destination address prefix. </paragraph>
  </description>
</field>

<field name="ipTTL" dataType="unsigned8"
       group="ipHeader"
       elementId="192" applicability="all" status="current">
  <description>
    <paragraph>
    For IPv4, the value of the Information Element matches
    the value of the Time to Live (TTL) field in the IPv4 packet
    header.  For IPv6, the value of the Information Element
    matches the value of the Hop Limit field in the IPv6
    packet header.
    </paragraph>
  </description>
  <reference>
    <paragraph>
    See RFC 791 for the definition of the IPv4
    Time to Live field.
    See RFC 2460 for the definition of the IPv6
    Hop Limit field.
    </paragraph>
  </reference>
  <units>hops</units>
</field>

<field name="protocolIdentifier" dataType="unsigned8"
       group="ipHeader"
       dataTypeSemantics="identifier"
       elementId="4" applicability="all" status="current">
  <description>
    <paragraph>
    The value of the protocol number in the IP packet header.
    The protocol number identifies the IP packet payload type.
    Protocol numbers are defined in the IANA Protocol Numbers
    registry.
       </paragraph>

    <paragraph>
    In Internet Protocol version 4 (IPv4), this is carried in the
    Protocol field.  In Internet Protocol version 6 (IPv6), this
```

```
          is carried in the Next Header field in the last extension
          header of the packet.
             </paragraph>
        </description>
        <reference>
          <paragraph>
          See RFC 791 for the specification of the IPv4
          protocol field.
          See RFC 2460 for the specification of the IPv6
          protocol field.
          See the list of protocol numbers assigned by IANA at
          http://www.iana.org/assignments/protocol-numbers.
          </paragraph>
        </reference>
      </field>

      <field name="nextHeaderIPv6" dataType="unsigned8"
             group="ipHeader"
             elementId="193" applicability="all" status="current">
        <description>
          <paragraph>
          The value of the Next Header field of the IPv6 header.
          The value identifies the type of the following IPv6
          extension header or of the following IP payload.
          Valid values are defined in the IANA
          Protocol Numbers registry.
          </paragraph>
        </description>
        <reference>
          <paragraph>
          See RFC 2460 for the definition of the IPv6
          Next Header field.
          See the list of protocol numbers assigned by IANA at
          http://www.iana.org/assignments/protocol-numbers.
          </paragraph>
        </reference>
      </field>

      <field name="ipDiffServCodePoint" dataType="unsigned8"
             group="ipHeader"
             dataTypeSemantics="identifier"
             elementId="195" applicability="all" status="current">
        <description>
          <paragraph>
          The value of a Differentiated Services Code Point (DSCP)
          encoded in the Differentiated Services field.  The
          Differentiated Services field spans the most significant
          6 bits of the IPv4 TOS field or the IPv6 Traffic Class
```

```
         field, respectively.
         </paragraph>
         <paragraph>
         This Information Element encodes only the 6 bits of the
         Differentiated Services field.  Therefore, its value may
         range from 0 to 63.
         </paragraph>
       </description>
       <reference>
         <paragraph>
         See RFC 3260 for the definition of the
         Differentiated Services field.
         See RFC 1812 (Section 5.3.2) and RFC 791 for the definition
         of the IPv4 TOS field.  See RFC 2460 for the definition of
         the IPv6 Traffic Class field.
         </paragraph>
       </reference>
       <range>0-63</range>
     </field>

     <field name="ipPrecedence" dataType="unsigned8"
            group="ipHeader"
            dataTypeSemantics="identifier"
            elementId="196" applicability="all" status="current">
       <description>
         <paragraph>
         The value of the IP Precedence.  The IP Precedence value
         is encoded in the first 3 bits of the IPv4 TOS field
         or the IPv6 Traffic Class field, respectively.
         </paragraph>
         <paragraph>
         This Information Element encodes only these 3 bits.
         Therefore, its value may range from 0 to 7.
         </paragraph>
       </description>
       <reference>
         <paragraph>
         See RFC 1812 (Section 5.3.3) and RFC 791
         for the definition of the IP Precedence.
         See RFC 1812 (Section 5.3.2) and RFC 791
         for the definition of the IPv4 TOS field.
         See RFC 2460 for the definition of the IPv6
         Traffic Class field.
         </paragraph>
       </reference>
       <range>0-7</range>
     </field>
```

```
<field name="ipClassOfService" dataType="unsigned8"
       group="ipHeader"
       dataTypeSemantics="identifier"
       elementId="5" applicability="all" status="current">
  <description>
    <paragraph>
    For IPv4 packets, this is the value of the TOS field in
    the IPv4 packet header.  For IPv6 packets, this is the
    value of the Traffic Class field in the IPv6 packet header.
    </paragraph>
  </description>
  <reference>
    <paragraph>
    See RFC 1812 (Section 5.3.2) and RFC 791
    for the definition of the IPv4 TOS field.
    See RFC 2460 for the definition of the IPv6
    Traffic Class field.
    </paragraph>
  </reference>
</field>

<field name="postIpClassOfService" dataType="unsigned8"
       group="ipHeader"
       dataTypeSemantics="identifier"
       elementId="55" applicability="all" status="current">
  <description>
    <paragraph>
    The definition of this Information Element is identical
    to the definition of Information Element
    'ipClassOfService', except that it reports a
    potentially modified value caused by a middlebox
    function after the packet passed the Observation Point.
    </paragraph>
  </description>
  <reference>
    <paragraph>
    See RFC 791 for the definition of the IPv4
    TOS field.
    See RFC 2460 for the definition of the IPv6
    Traffic Class field.
    See RFC 3234 for the definition of middleboxes.
    </paragraph>
  </reference>
</field>

<field name="flowLabelIPv6" dataType="unsigned32"
       group="ipHeader"
       dataTypeSemantics="identifier"
```

```
             elementId="31" applicability="all" status="current">
        <description>
          <paragraph>
          The value of the IPv6 Flow Label field in the IP packet header.
          </paragraph>
        </description>
        <reference>
          <paragraph>
          See RFC 2460 for the definition of the
          Flow Label field in the IPv6 packet header.
          </paragraph>
        </reference>
      </field>

      <field name="isMulticast" dataType="unsigned8"
             group="ipHeader"
             dataTypeSemantics="flags"
             elementId="206" applicability="data" status="current">
        <description>
          <paragraph>
            If the IP destination address is not a reserved multicast
            address, then the value of all bits of the octet (including
            the reserved ones) is zero.
          </paragraph>
          <paragraph>
            The first bit of this octet is set to 1 if the Version
            field of the IP header has the value 4 and if the
            Destination Address field contains a reserved multicast
            address in the range from 224.0.0.0 to 239.255.255.255.
            Otherwise, this bit is set to 0.
          </paragraph>
          <paragraph>
            The second and third bits of this octet are reserved for
            future use.
          </paragraph>
          <paragraph>
            The remaining bits of the octet are only set to values
            other than zero if the IP Destination Address is a
            reserved IPv6 multicast address.  Then the fourth bit
            of the octet is set to the value of the T flag in the
            IPv6 multicast address and the remaining four bits are
            set to the value of the scope field in the IPv6
            multicast address.
          </paragraph>
          <artwork>
              0      1      2      3      4      5      6      7
           +------+------+------+------+------+------+------+------+
           |   IPv6 multicast scope    |  T   | RES. | RES. | MCv4 |
```

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

          Bit  0:    set to 1 if IPv4 multicast
          Bits 1-2:  reserved for future use
          Bit  4:    set to value of T flag, if IPv6 multicast
          Bits 4-7:  set to value of multicast scope if IPv6 multicast
        </artwork>
      </description>
      <reference>
        <paragraph>
        See RFC 1112 for the specification of reserved
        IPv4 multicast addresses.
        See RFC 4291 for the specification of reserved
        IPv6 multicast addresses and the definition of the T flag
        and the IPv6 multicast scope.
        </paragraph>
      </reference>
    </field>

    <field name="fragmentIdentification" dataType="unsigned32"
           group="ipHeader"
           dataTypeSemantics="identifier"
           elementId="54" applicability="data" status="current">
      <description>
        <paragraph>
        The value of the Identification field
        in the IPv4 packet header or in the IPv6 Fragment header,
        respectively.  The value is 0 for IPv6 if there is
        no fragment header.
        </paragraph>
      </description>
      <reference>
        <paragraph>
        See RFC 791 for the definition of the IPv4
        Identification field.
        See RFC 2460 for the definition of the
        Identification field in the IPv6 Fragment header.
        </paragraph>
      </reference>
    </field>

    <field name="fragmentOffset" dataType="unsigned16"
           group="ipHeader"
           dataTypeSemantics="identifier"
           elementId="88" applicability="all" status="current">
      <description>
        <paragraph>
        The value of the IP fragment offset field in the
```

```
          IPv4 packet header or the IPv6 Fragment header,
          respectively.  The value is 0 for IPv6 if there is
          no fragment header.
          </paragraph>
        </description>
        <reference>
          <paragraph>
          See RFC 791 for the specification of the
          fragment offset in the IPv4 header.
          See RFC 2460 for the specification of the
          fragment offset in the IPv6 Fragment header.
          </paragraph>
        </reference>
      </field>

      <field name="fragmentFlags" dataType="unsigned8"
             group="ipHeader"
             dataTypeSemantics="flags"
             elementId="197" applicability="all" status="current">
        <description>
          <paragraph>
            Fragmentation properties indicated by flags in the IPv4
            packet header or the IPv6 Fragment header, respectively.
          </paragraph>
          <artwork>

          Bit 0:    (RS) Reserved.
                    The value of this bit MUST be 0 until specified
                    otherwise.
          Bit 1:    (DF) 0 = May Fragment,  1 = Don't Fragment.
                    Corresponds to the value of the DF flag in the
                    IPv4 header.  Will always be 0 for IPv6 unless
                    a "don't fragment" feature is introduced to IPv6.
          Bit 2:    (MF) 0 = Last Fragment, 1 = More Fragments.
                    Corresponds to the MF flag in the IPv4 header
                    or to the M flag in the IPv6 Fragment header,
                    respectively.  The value is 0 for IPv6 if there
                    is no fragment header.
          Bits 3-7: (DC) Don't Care.
                    The values of these bits are irrelevant.

              0   1   2   3   4   5   6   7
            +---+---+---+---+---+---+---+---+
            | R | D | M | D | D | D | D | D |
            | S | F | F | C | C | C | C | C |
            +---+---+---+---+---+---+---+---+
          </artwork>
        </description>
```

```
        <reference>
          <paragraph>
          See RFC 791 for the specification of the IPv4
          fragment flags.
          See RFC 2460 for the specification of the IPv6
          Fragment header.
          </paragraph>
        </reference>
      </field>

      <field name="ipHeaderLength" dataType="unsigned8"
             group="ipHeader"
             elementId="189" applicability="all" status="current">
        <description>
          <paragraph>
          The length of the IP header.  For IPv6, the value of this
          Information Element is 40.
          </paragraph>
        </description>
        <reference>
          <paragraph>
          See RFC 791 for the specification of the
          IPv4 header.
          See RFC 2460 for the specification of the
          IPv6 header.
          </paragraph>
        </reference>
        <units>octets</units>
      </field>

      <field name="ipv4IHL" dataType="unsigned8"
             group="ipHeader"
             elementId="207" applicability="all" status="current">
        <description>
          <paragraph>
          The value of the Internet Header Length (IHL) field in
          the IPv4 header.  It specifies the length of the header
          in units of 4 octets.  Please note that its unit is
          different from most of the other Information Elements
          reporting length values.
          </paragraph>
        </description>
        <reference>
          <paragraph>
          See RFC 791 for the specification of the
          IPv4 header.
          </paragraph>
        </reference>
```

```
    <units>4 octets</units>
  </field>

  <field name="totalLengthIPv4" dataType="unsigned16"
        group="ipHeader"
        elementId="190" applicability="all" status="current">
    <description>
      <paragraph>
      The total length of the IPv4 packet.
      </paragraph>
    </description>
    <reference>
      <paragraph>
      See RFC 791 for the specification of the
      IPv4 total length.
      </paragraph>
    </reference>
    <units>octets</units>
  </field>

  <field name="ipTotalLength" dataType="unsigned64"
        group="ipHeader"
        elementId="224" applicability="all" status="current">
    <description>
      <paragraph>
      The total length of the IP packet.
      </paragraph>
    </description>
    <reference>
      <paragraph>
      See RFC 791 for the specification of the
      IPv4 total length.
      See RFC 2460 for the specification of the
      IPv6 payload length.
      See RFC 2675 for the specification of the
      IPv6 jumbo payload length.
      </paragraph>
    </reference>
    <units>octets</units>
  </field>

  <field name="payloadLengthIPv6" dataType="unsigned16"
        group="ipHeader"
        elementId="191" applicability="all" status="current">
    <description>
      <paragraph>
      This Information Element reports the value of the Payload
      Length field in the IPv6 header.  Note that IPv6 extension
```

```
         headers belong to the payload.  Also note that in case of a
         jumbo payload option the value of the Payload Length field in
         the IPv6 header is zero and so will be the value reported
         by this Information Element.
         </paragraph>
       </description>
       <reference>
         <paragraph>
         See RFC 2460 for the specification of the
         IPv6 payload length.
         See RFC 2675 for the specification of the
         IPv6 jumbo payload option.
         </paragraph>
       </reference>
       <units>octets</units>
     </field>

     <field name="sourceTransportPort" dataType="unsigned16"
            group="transportHeader"
            dataTypeSemantics="identifier"
            elementId="7" applicability="all" status="current">
       <description>
         <paragraph>
         The source port identifier in the transport header.
         For the transport protocols UDP, TCP, and SCTP, this is the
         source port number given in the respective header.  This
         field MAY also be used for future transport protocols that
         have 16-bit source port identifiers.
         </paragraph>
       </description>
       <reference>
         <paragraph>
         See RFC 768 for the definition of the UDP
         source port field.
         See RFC 793 for the definition of the TCP
         source port field.
         See RFC 4960 for the definition of SCTP.
         </paragraph>
         <paragraph>
         Additional information on defined UDP and TCP port numbers can
         be found at http://www.iana.org/assignments/port-numbers.
         </paragraph>
       </reference>
     </field>

     <field name="destinationTransportPort" dataType="unsigned16"
            group="transportHeader"
            dataTypeSemantics="identifier"
```

```
           elementId="11" applicability="all" status="current">
    <description>
      <paragraph>
      The destination port identifier in the transport header.
      For the transport protocols UDP, TCP, and SCTP, this is the
      destination port number given in the respective header.
      This field MAY also be used for future transport protocols
      that have 16-bit destination port identifiers.
      </paragraph>
    </description>
    <reference>
      <paragraph>
      See RFC 768 for the definition of the UDP
      destination port field.
      See RFC 793 for the definition of the TCP
      destination port field.
      See RFC 4960 for the definition of SCTP.
      </paragraph>
      <paragraph>
      Additional information on defined UDP and TCP port numbers can
      be found at http://www.iana.org/assignments/port-numbers.
      </paragraph>
    </reference>
  </field>

  <field name="udpSourcePort" dataType="unsigned16"
         group="transportHeader"
         dataTypeSemantics="identifier"
         elementId="180" applicability="all" status="current">
    <description>
      <paragraph>
      The source port identifier in the UDP header.
      </paragraph>
    </description>
    <reference>
      <paragraph>
      See RFC 768 for the definition of the
      UDP source port field.
      Additional information on defined UDP port numbers can
      be found at http://www.iana.org/assignments/port-numbers.
      </paragraph>
    </reference>
  </field>

  <field name="udpDestinationPort" dataType="unsigned16"
         group="transportHeader"
         dataTypeSemantics="identifier"
         elementId="181" applicability="all" status="current">
```

```
        <description>
          <paragraph>
          The destination port identifier in the UDP header.
          </paragraph>
        </description>
        <reference>
          <paragraph>
          See RFC 768 for the definition of the
          UDP destination port field.
          Additional information on defined UDP port numbers can
          be found at http://www.iana.org/assignments/port-numbers.
          </paragraph>
        </reference>
      </field>

      <field name="udpMessageLength" dataType="unsigned16"
             group="transportHeader"
             elementId="205" applicability="all" status="current">
        <description>
          <paragraph>
          The value of the Length field in the UDP header.
          </paragraph>
        </description>
        <reference>
          <paragraph>
          See RFC 768 for the specification of the
          UDP header.
          </paragraph>
        </reference>
        <units>octets</units>
      </field>

      <field name="tcpSourcePort" dataType="unsigned16"
             group="transportHeader"
             dataTypeSemantics="identifier"
             elementId="182" applicability="all" status="current">
        <description>
          <paragraph>
          The source port identifier in the TCP header.
          </paragraph>
        </description>
        <reference>
          <paragraph>
          See RFC 793 for the definition of the TCP
          source port field.
          Additional information on defined TCP port numbers can
          be found at http://www.iana.org/assignments/port-numbers.
          </paragraph>
```

```
      </reference>
    </field>

    <field name="tcpDestinationPort" dataType="unsigned16"
           group="transportHeader"
           dataTypeSemantics="identifier"
           elementId="183" applicability="all" status="current">
      <description>
        <paragraph>
        The destination port identifier in the TCP header.
        </paragraph>
      </description>
      <reference>
        <paragraph>
        See RFC 793 for the definition of the TCP
        destination port field.
        Additional information on defined TCP port numbers can
        be found at http://www.iana.org/assignments/port-numbers.
        </paragraph>
      </reference>
    </field>

    <field name="tcpSequenceNumber" dataType="unsigned32"
           group="transportHeader"
           elementId="184" applicability="all" status="current">
      <description>
        <paragraph>
        The sequence number in the TCP header.
        </paragraph>
      </description>
      <reference>
        <paragraph>
        See RFC 793 for the definition of the TCP
        sequence number.
        </paragraph>
      </reference>
    </field>

    <field name="tcpAcknowledgementNumber" dataType="unsigned32"
           group="transportHeader"
           elementId="185" applicability="all" status="current">
      <description>
        <paragraph>
        The acknowledgement number in the TCP header.
        </paragraph>
      </description>
      <reference>
        <paragraph>
```

```
         See RFC 793 for the definition of the TCP
         acknowledgement number.
         </paragraph>
       </reference>
     </field>

     <field name="tcpWindowSize" dataType="unsigned16"
            group="transportHeader"
            elementId="186" applicability="all" status="current">
       <description>
         <paragraph>
         The window field in the TCP header.
         If the TCP window scale is supported,
         then TCP window scale must be known
         to fully interpret the value of this information.
         </paragraph>
       </description>
       <reference>
         <paragraph>
         See RFC 793 for the definition of the TCP window field.
         See RFC 1323 for the definition of the TCP window scale.
         </paragraph>
       </reference>
     </field>

     <field name="tcpWindowScale" dataType="unsigned16"
            group="transportHeader"
            elementId="238" applicability="all" status="current">
       <description>
         <paragraph>
         The scale of the window field in the TCP header.
         </paragraph>
       </description>
       <reference>
         <paragraph>
         See RFC 1323 for the definition of the TCP window scale.
         </paragraph>
       </reference>
     </field>

     <field name="tcpUrgentPointer" dataType="unsigned16"
            group="transportHeader"
            elementId="187" applicability="all" status="current">
       <description>
         <paragraph>
         The urgent pointer in the TCP header.
         </paragraph>
       </description>
```

```
      <reference>
        <paragraph>
        See RFC 793 for the definition of the TCP
        urgent pointer.
        </paragraph>
      </reference>
    </field>

    <field name="tcpHeaderLength" dataType="unsigned8"
           group="transportHeader"
           elementId="188" applicability="all" status="current">
      <description>
        <paragraph>
        The length of the TCP header.  Note that the value of this
        Information Element is different from the value of the Data
        Offset field in the TCP header.  The Data Offset field
        indicates the length of the TCP header in units of 4 octets.
        This Information Elements specifies the length of the TCP
        header in units of octets.
        </paragraph>
      </description>
      <reference>
        <paragraph>
        See RFC 793 for the definition of the
        TCP header.
        </paragraph>
      </reference>
      <units>octets</units>
    </field>

    <field name="icmpTypeCodeIPv4" dataType="unsigned16"
           group="transportHeader"
           dataTypeSemantics="identifier"
           elementId="32" applicability="all" status="current">
      <description>
        <paragraph>
        Type and Code of the IPv4 ICMP message.  The combination of
        both values is reported as (ICMP type * 256) + ICMP code.
        </paragraph>
      </description>
      <reference>
        <paragraph>
        See RFC 792 for the definition of the IPv4 ICMP
           type and code fields.
        </paragraph>
      </reference>
    </field>
```

```
<field name="icmpTypeIPv4" dataType="unsigned8"
       group="transportHeader"
       dataTypeSemantics="identifier"
       elementId="176" applicability="all" status="current">
  <description>
    <paragraph>
    Type of the IPv4 ICMP message.
    </paragraph>
  </description>
  <reference>
    <paragraph>
    See RFC 792 for the definition of the IPv4 ICMP
    type field.
    </paragraph>
  </reference>
</field>

<field name="icmpCodeIPv4" dataType="unsigned8"
       group="transportHeader"
       dataTypeSemantics="identifier"
       elementId="177" applicability="all" status="current">
  <description>
    <paragraph>
    Code of the IPv4 ICMP message.
    </paragraph>
  </description>
  <reference>
    <paragraph>
    See RFC 792 for the definition of the IPv4
    ICMP code field.
    </paragraph>
  </reference>
</field>

<field name="icmpTypeCodeIPv6" dataType="unsigned16"
       group="transportHeader"
       dataTypeSemantics="identifier"
       elementId="139" applicability="all" status="current">
  <description>
    <paragraph>
    Type and Code of the IPv6 ICMP message.  The combination of
    both values is reported as (ICMP type * 256) + ICMP code.
    </paragraph>
  </description>
  <reference>
    <paragraph>
    See RFC 4443 for the definition of the IPv6
    ICMP type and code fields.
```

```
        &lt;/paragraph&gt;
      &lt;/reference&gt;
    &lt;/field&gt;

    &lt;field name="icmpTypeIPv6" dataType="unsigned8"
           group="transportHeader"
           dataTypeSemantics="identifier"
           elementId="178" applicability="all" status="current"&gt;
      &lt;description&gt;
        &lt;paragraph&gt;
        Type of the IPv6 ICMP message.
        &lt;/paragraph&gt;
      &lt;/description&gt;
      &lt;reference&gt;
        &lt;paragraph&gt;
        See RFC 4443 for the definition of the IPv6
        ICMP type field.
        &lt;/paragraph&gt;
      &lt;/reference&gt;
    &lt;/field&gt;

    &lt;field name="icmpCodeIPv6" dataType="unsigned8"
           group="transportHeader"
           dataTypeSemantics="identifier"
           elementId="179" applicability="all" status="current"&gt;
      &lt;description&gt;
        &lt;paragraph&gt;
        Code of the IPv6 ICMP message.
        &lt;/paragraph&gt;
      &lt;/description&gt;
      &lt;reference&gt;
        &lt;paragraph&gt;
        See RFC 4443 for the definition of the IPv6
        ICMP code field.
        &lt;/paragraph&gt;
      &lt;/reference&gt;
    &lt;/field&gt;

    &lt;field name="igmpType" dataType="unsigned8"
           group="transportHeader"
           dataTypeSemantics="identifier"
           elementId="33" applicability="all" status="current"&gt;
      &lt;description&gt;
        &lt;paragraph&gt;
        The type field of the IGMP message.
        &lt;/paragraph&gt;
      &lt;/description&gt;
      &lt;reference&gt;
```

```
          <paragraph>
          See RFC 3376 for the definition of the IGMP
          type field.
          </paragraph>
        </reference>
      </field>

      <field name="sourceMacAddress" dataType="macAddress"
             group="subIpHeader"
             dataTypeSemantics="identifier"
             elementId="56" applicability="data" status="current">
        <description>
          <paragraph>
            The IEEE 802 source MAC address field.
          </paragraph>
        </description>
        <reference>
          <paragraph>
          See IEEE.802-3.2002.
          </paragraph>
        </reference>
      </field>

      <field name="postSourceMacAddress" dataType="macAddress"
             group="subIpHeader"
             dataTypeSemantics="identifier"
             elementId="81" applicability="data" status="current">
        <description>
          <paragraph>
          The definition of this Information Element is identical
          to the definition of Information Element
          'sourceMacAddress', except that it reports a
          potentially modified value caused by a middlebox
          function after the packet passed the Observation Point.
          </paragraph>
        </description>
        <reference>
          <paragraph>
          See IEEE.802-3.2002.
          </paragraph>
        </reference>
      </field>

      <field name="vlanId" dataType="unsigned16"
             group="subIpHeader"
             dataTypeSemantics="identifier"
             elementId="58" applicability="data" status="current">
        <description>
```

```
        <paragraph>
          The IEEE 802.1Q VLAN identifier (VID) extracted from the Tag
          Control Information field that was attached to the IP packet.
        </paragraph>
      </description>
      <reference>
        <paragraph>
        See IEEE.802-1Q.2003.
        </paragraph>
      </reference>
    </field>

    <field name="postVlanId" dataType="unsigned16"
           group="subIpHeader"
           dataTypeSemantics="identifier"
           elementId="59" applicability="data" status="current">
      <description>
        <paragraph>
        The definition of this Information Element is identical
        to the definition of Information Element
        'vlanId', except that it reports a
        potentially modified value caused by a middlebox
        function after the packet passed the Observation Point.
        </paragraph>
      </description>
      <reference>
        <paragraph>
        See IEEE.802-1Q.2003.
        </paragraph>
      </reference>
    </field>

    <field name="destinationMacAddress" dataType="macAddress"
           group="subIpHeader"
           dataTypeSemantics="identifier"
           elementId="80" applicability="data" status="current">
      <description>
        <paragraph>
          The IEEE 802 destination MAC address field.
        </paragraph>
      </description>
      <reference>
        <paragraph>
        See IEEE.802-3.2002.
        </paragraph>
      </reference>
    </field>
```

```
     <field name="postDestinationMacAddress" dataType="macAddress"
            group="subIpHeader"
            dataTypeSemantics="identifier"
            elementId="57" applicability="data" status="current">
       <description>
         <paragraph>
         The definition of this Information Element is identical
         to the definition of Information Element
         'destinationMacAddress', except that it reports a
         potentially modified value caused by a middlebox
         function after the packet passed the Observation Point.
         </paragraph>
       </description>
       <reference>
         <paragraph>
         See IEEE.802-3.2002.
         </paragraph>
       </reference>
     </field>

     <field name="wlanChannelId" dataType="unsigned8"
            group="subIpHeader"
            dataTypeSemantics="identifier"
            elementId="146" applicability="data" status="current">
       <description>
         <paragraph>
           The identifier of the 802.11 (Wi-Fi) channel used.
         </paragraph>
       </description>
       <reference>
         <paragraph>
         See IEEE.802-11.1999.
         </paragraph>
       </reference>
     </field>

     <field name="wlanSSID" dataType="string"
            group="subIpHeader"
            elementId="147" applicability="data" status="current">
       <description>
         <paragraph>
           The Service Set IDentifier (SSID) identifying an 802.11
           (Wi-Fi) network used.  According to IEEE.802-11.1999, the
           SSID is encoded into a string of up to 32 characters.
         </paragraph>
       </description>
       <reference>
         <paragraph>
```

```
         See IEEE.802-11.1999.
         </paragraph>
       </reference>
     </field>

     <field name="mplsTopLabelTTL" dataType="unsigned8"
            group="subIpHeader"
            elementId="200" applicability="all" status="current">
       <description>
         <paragraph>
         The TTL field from the top MPLS label stack entry,
         i.e., the last label that was pushed.
         </paragraph>
       </description>
       <reference>
         <paragraph>
         See RFC 3032 for the specification of the
         TTL field.
         </paragraph>
       </reference>
       <units>hops</units>
     </field>

     <field name="mplsTopLabelExp" dataType="unsigned8"
            group="subIpHeader"
            dataTypeSemantics="flags"
            elementId="203" applicability="all" status="current">
       <description>
         <paragraph>
         The Exp field from the top MPLS label stack entry,
         i.e., the last label that was pushed.
         </paragraph>
         <artwork>
         Bits 0-4:  Don't Care, value is irrelevant.
         Bits 5-7:  MPLS Exp field.

             0   1   2   3   4   5   6   7
           +---+---+---+---+---+---+---+---+
           |     don't care    |    Exp    |
           +---+---+---+---+---+---+---+---+
         </artwork>
       </description>
       <reference>
         <paragraph>
         See RFC 3032 for the specification of the Exp field.
         See RFC 3270 for usage of the Exp field.
         </paragraph>
       </reference>
```

```
     </field>

     <field name="postMplsTopLabelExp" dataType="unsigned8"
            group="subIpHeader"
            dataTypeSemantics="flags"
            elementId="237" applicability="all" status="current">
       <description>
         <paragraph>
         The definition of this Information Element is identical to the
         definition of Information Element 'mplsTopLabelExp', except
         that it reports a potentially modified value caused by a
         middlebox function after the packet passed the Observation
         Point.
         </paragraph>
       </description>
       <reference>
         <paragraph>
         See RFC 3032 for the specification of the Exp field.
         See RFC 3270 for usage of the Exp field.
         </paragraph>
       </reference>
     </field>

     <field name="mplsLabelStackDepth" dataType="unsigned32"
            group="subIpHeader"
            elementId="202" applicability="all" status="current">
       <description>
         <paragraph>
         The number of labels in the MPLS label stack.
         </paragraph>
       </description>
       <reference>
         <paragraph>
         See RFC 3032 for the specification of
         the MPLS label stack.
         </paragraph>
       </reference>
       <units>label stack entries</units>
     </field>

     <field name="mplsLabelStackLength" dataType="unsigned32"
            group="subIpHeader"
            elementId="201" applicability="all" status="current">
       <description>
         <paragraph>
         The length of the MPLS label stack in units of octets.
         </paragraph>
       </description>
```

```
        <reference>
          <paragraph>
          See RFC 3032 for the specification of
          the MPLS label stack.
          </paragraph>
        </reference>
        <units>octets</units>
      </field>

      <field name="mplsPayloadLength" dataType="unsigned32"
             group="subIpHeader"
             elementId="194" applicability="all" status="current">
        <description>
          <paragraph>
          The size of the MPLS packet without the label stack.
          </paragraph>
        </description>
        <reference>
          <paragraph>
          See RFC 3031 for the specification of
          MPLS packets.
          See RFC 3032 for the specification of
          the MPLS label stack.
          </paragraph>
        </reference>
        <units>octets</units>
      </field>

      <field name="mplsTopLabelStackSection" dataType="octetArray"
             group="subIpHeader"
             dataTypeSemantics="identifier"
             elementId="70" applicability="all" status="current">
        <description>
          <paragraph>
          The Label, Exp, and S fields from the top MPLS label
          stack entry, i.e., from the last label that was pushed.
          </paragraph>
          <paragraph>
          The size of this Information Element is 3 octets.
          </paragraph>
          <artwork>
        0                   1                   2
        0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                Label                  | Exp |S|
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

       Label:  Label Value, 20 bits
```

```
      Exp:    Experimental Use, 3 bits
      S:      Bottom of Stack, 1 bit
        </artwork>
      </description>
      <reference>
        <paragraph>
        See RFC 3032.
        </paragraph>
      </reference>
    </field>

    <field name="mplsLabelStackSection2" dataType="octetArray"
           group="subIpHeader"
           dataTypeSemantics="identifier"
           elementId="71" applicability="all" status="current">
      <description>
        <paragraph>
        The Label, Exp, and S fields from the label stack entry that
        was pushed immediately before the label stack entry that would
        be reported by mplsTopLabelStackSection.  See the definition of
        mplsTopLabelStackSection for further details.
        </paragraph>
        <paragraph>
        The size of this Information Element is 3 octets.
        </paragraph>
      </description>
      <reference>
        <paragraph>
        See RFC 3032.
        </paragraph>
      </reference>
    </field>

    <field name="mplsLabelStackSection3" dataType="octetArray"
           group="subIpHeader"
           dataTypeSemantics="identifier"
           elementId="72" applicability="all" status="current">
      <description>
        <paragraph>
        The Label, Exp, and S fields from the label stack entry that
        was pushed immediately before the label stack entry that would
        be reported by mplsLabelStackSection2.  See the definition of
        mplsTopLabelStackSection for further details.
        </paragraph>
        <paragraph>
        The size of this Information Element is 3 octets.
        </paragraph>
      </description>
```

```
        <reference>
          <paragraph>
          See RFC 3032.
          </paragraph>
        </reference>
      </field>

      <field name="mplsLabelStackSection4" dataType="octetArray"
             group="subIpHeader"
             dataTypeSemantics="identifier"
             elementId="73" applicability="all" status="current">
        <description>
          <paragraph>
          The Label, Exp, and S fields from the label stack entry that
          was pushed immediately before the label stack entry that would
          be reported by mplsLabelStackSection3.  See the definition of
          mplsTopLabelStackSection for further details.
          </paragraph>
          <paragraph>
          The size of this Information Element is 3 octets.
          </paragraph>
        </description>
        <reference>
          <paragraph>
          See RFC 3032.
          </paragraph>
        </reference>
      </field>

      <field name="mplsLabelStackSection5" dataType="octetArray"
             group="subIpHeader"
             dataTypeSemantics="identifier"
             elementId="74" applicability="all" status="current">
        <description>
          <paragraph>
          The Label, Exp, and S fields from the label stack entry that
          was pushed immediately before the label stack entry that would
          be reported by mplsLabelStackSection4.  See the definition of
          mplsTopLabelStackSection for further details.
          </paragraph>
          <paragraph>
          The size of this Information Element is 3 octets.
          </paragraph>
        </description>
        <reference>
          <paragraph>
          See RFC 3032.
          </paragraph>
```

```
    </reference>
  </field>

  <field name="mplsLabelStackSection6" dataType="octetArray"
         group="subIpHeader"
         dataTypeSemantics="identifier"
         elementId="75" applicability="all" status="current">
    <description>
      <paragraph>
      The Label, Exp, and S fields from the label stack entry that
      was pushed immediately before the label stack entry that would
      be reported by mplsLabelStackSection5.  See the definition of
      mplsTopLabelStackSection for further details.
      </paragraph>
      <paragraph>
      The size of this Information Element is 3 octets.
      </paragraph>
    </description>
    <reference>
      <paragraph>
      See RFC 3032.
      </paragraph>
    </reference>
  </field>

  <field name="mplsLabelStackSection7" dataType="octetArray"
         group="subIpHeader"
         dataTypeSemantics="identifier"
         elementId="76" applicability="all" status="current">
    <description>
      <paragraph>
      The Label, Exp, and S fields from the label stack entry that
      was pushed immediately before the label stack entry that would
      be reported by mplsLabelStackSection6.  See the definition of
      mplsTopLabelStackSection for further details.
      </paragraph>
      <paragraph>
      The size of this Information Element is 3 octets.
      </paragraph>
    </description>
    <reference>
      <paragraph>
      See RFC 3032.
      </paragraph>
    </reference>
  </field>

  <field name="mplsLabelStackSection8" dataType="octetArray"
```

```
           group="subIpHeader"
           dataTypeSemantics="identifier"
           elementId="77" applicability="all" status="current">
    <description>
      <paragraph>
      The Label, Exp, and S fields from the label stack entry that
      was pushed immediately before the label stack entry that would
      be reported by mplsLabelStackSection7.  See the definition of
      mplsTopLabelStackSection for further details.
      </paragraph>
      <paragraph>
      The size of this Information Element is 3 octets.
      </paragraph>
    </description>
    <reference>
      <paragraph>
      See RFC 3032.
      </paragraph>
    </reference>
  </field>

  <field name="mplsLabelStackSection9" dataType="octetArray"
         group="subIpHeader"
         dataTypeSemantics="identifier"
         elementId="78" applicability="all" status="current">
    <description>
      <paragraph>
      The Label, Exp, and S fields from the label stack entry that
      was pushed immediately before the label stack entry that would
      be reported by mplsLabelStackSection8.  See the definition of
      mplsTopLabelStackSection for further details.
      </paragraph>
      <paragraph>
      The size of this Information Element is 3 octets.
      </paragraph>
    </description>
    <reference>
      <paragraph>
      See RFC 3032.
      </paragraph>
    </reference>
  </field>

  <field name="mplsLabelStackSection10" dataType="octetArray"
         group="subIpHeader"
         dataTypeSemantics="identifier"
         elementId="79" applicability="all" status="current">
    <description>
```

```
        <paragraph>
        The Label, Exp, and S fields from the label stack entry that
        was pushed immediately before the label stack entry that would
        be reported by mplsLabelStackSection9.  See the definition of
        mplsTopLabelStackSection for further details.
        </paragraph>
        <paragraph>
        The size of this Information Element is 3 octets.
        </paragraph>
      </description>
      <reference>
        <paragraph>
        See RFC 3032.
        </paragraph>
      </reference>
    </field>

    <field name="ipPayloadLength" dataType="unsigned32"
           group="derived"
           elementId="204" applicability="all" status="current">
      <description>
        <paragraph>
        The effective length of the IP payload.
        </paragraph>
        <paragraph>
        For IPv4 packets, the value of this Information Element is
        the difference between the total length of the IPv4 packet
        (as reported by Information Element totalLengthIPv4) and the
        length of the IPv4 header (as reported by Information Element
        headerLengthIPv4).
        </paragraph>
        <paragraph>
        For IPv6, the value of the Payload Length field
        in the IPv6 header is reported except in the case that
        the value of this field is zero and that there is a valid
        jumbo payload option.  In this case, the value of the
        Jumbo Payload Length field in the jumbo payload option
        is reported.
        </paragraph>
      </description>
      <reference>
        <paragraph>
        See RFC 791 for the specification of
        IPv4 packets.
        See RFC 2460 for the specification of the
        IPv6 payload length.
        See RFC 2675 for the specification of the
        IPv6 jumbo payload length.
```

```
        </paragraph>
      </reference>
      <units>octets</units>
    </field>

    <field name="ipNextHopIPv4Address" dataType="ipv4Address"
           group="derived"
           dataTypeSemantics="identifier"
           elementId="15" applicability="data" status="current">
      <description>
        <paragraph>
        The IPv4 address of the next IPv4 hop.
        </paragraph>
      </description>
    </field>

    <field name="ipNextHopIPv6Address" dataType="ipv6Address"
           group="derived"
           dataTypeSemantics="identifier"
           elementId="62" applicability="data" status="current">
      <description>
        <paragraph>
        The IPv6 address of the next IPv6 hop.
        </paragraph>
      </description>
    </field>

    <field name="bgpSourceAsNumber" dataType="unsigned32"
           group="derived"
           dataTypeSemantics="identifier"
           elementId="16" applicability="all" status="current">
      <description>
        <paragraph>
        The autonomous system (AS) number of the source IP address.
        If AS path information for this Flow is only available as
        an unordered AS set (and not as an ordered AS sequence),
        then the value of this Information Element is 0.
        </paragraph>
      </description>
      <reference>
        <paragraph>
        See RFC 4271 for a description of BGP-4, and see RFC 1930
        for the definition of the AS number.
        </paragraph>
      </reference>
    </field>

    <field name="bgpDestinationAsNumber" dataType="unsigned32"
```

```
            group="derived"
            dataTypeSemantics="identifier"
            elementId="17" applicability="all" status="current">
       <description>
         <paragraph>
         The autonomous system (AS) number of the destination IP
         address.  If AS path information for this Flow is only
         available as an unordered AS set (and not as an ordered AS
         sequence), then the value of this Information Element is 0.
         </paragraph>
       </description>
       <reference>
         <paragraph>
         See RFC 4271 for a description of BGP-4, and see RFC 1930 for
            the definition of the AS number.
         </paragraph>
       </reference>
     </field>

     <field name="bgpNextAdjacentAsNumber" dataType="unsigned32"
            group="derived"
            dataTypeSemantics="identifier"
            elementId="128" applicability="all" status="current">
       <description>
         <paragraph>
         The autonomous system (AS) number of the first AS in the AS
         path to the destination IP address.  The path is deduced
         by looking up the destination IP address of the Flow in the
         BGP routing information base.  If AS path information for
         this Flow is only available as an unordered AS set (and not
         as an ordered AS sequence), then the value of this Information
         Element is 0.
       </paragraph>
       </description>
       <reference>
         <paragraph>
         See RFC 4271 for a description of BGP-4, and
         see RFC 1930 for the definition of the AS number.
         </paragraph>
       </reference>
     </field>

     <field name="bgpPrevAdjacentAsNumber" dataType="unsigned32"
            group="derived"
            dataTypeSemantics="identifier"
            elementId="129" applicability="all" status="current">
       <description>
         <paragraph>
```

```
        The autonomous system (AS) number of the last AS in the AS
        path from the source IP address.  The path is deduced
        by looking up the source IP address of the Flow in the BGP
        routing information base.  If AS path information for this
        Flow is only available as an unordered AS set (and not as
        an ordered AS sequence), then the value of this Information
        Element is 0.  In case of BGP asymmetry, the
        bgpPrevAdjacentAsNumber might not be able to report the correct
        value.
      </paragraph>
      </description>
      <reference>
        <paragraph>
        See RFC 4271 for a description of BGP-4, and
        see RFC 1930 for the definition of the AS number.
        </paragraph>
      </reference>
    </field>

    <field name="bgpNextHopIPv4Address" dataType="ipv4Address"
           group="derived"
           dataTypeSemantics="identifier"
           elementId="18" applicability="all" status="current">
      <description>
        <paragraph>
        The IPv4 address of the next (adjacent) BGP hop.
        </paragraph>
      </description>
      <reference>
        <paragraph>
        See RFC 4271 for a description of BGP-4.
        </paragraph>
      </reference>
    </field>

    <field name="bgpNextHopIPv6Address" dataType="ipv6Address"
           group="derived"
           dataTypeSemantics="identifier"
           elementId="63" applicability="all" status="current">
      <description>
        <paragraph>
        The IPv6 address of the next (adjacent) BGP hop.
        </paragraph>
      </description>
      <reference>
        <paragraph>
        See RFC 4271 for a description of BGP-4.
        </paragraph>
```

```
    </reference>
  </field>

  <field name="mplsTopLabelType" dataType="unsigned8"
         group="derived"
         dataTypeSemantics="identifier"
         elementId="46" applicability="data" status="current">
    <description>
      <paragraph>
      This field identifies the control protocol that allocated the
      top-of-stack label.  Initial values for this field are
      listed below.  Further values may be assigned by IANA in
      the MPLS label type registry.
      </paragraph>
      <artwork>
     - 0x01 TE-MIDPT: Any TE tunnel mid-point or tail label
     - 0x02 Pseudowire: Any PWE3 or Cisco AToM based label
     - 0x03 VPN: Any label associated with VPN
     - 0x04 BGP: Any label associated with BGP or BGP routing
     - 0x05 LDP: Any label associated with dynamically assigned
                 labels using LDP
      </artwork>
    </description>
    <reference>
      <paragraph>
      See RFC 3031 for the MPLS label structure.
      See RFC 4364 for the association of MPLS labels
      with Virtual Private Networks (VPNs).
      See RFC 4271 for BGP and BGP routing.
      See RFC 5036 for Label Distribution Protocol (LDP).
      See the list of MPLS label types assigned by IANA at
      http://www.iana.org/assignments/mpls-label-values.
      </paragraph>
    </reference>
  </field>

  <field name="mplsTopLabelIPv4Address" dataType="ipv4Address"
         group="derived"
         dataTypeSemantics="identifier"
         elementId="47" applicability="data" status="current">
    <description>
      <paragraph>
        The IPv4 address of the system that the MPLS top label will
        cause this Flow to be forwarded to.
      </paragraph>
    </description>
    <reference>
      <paragraph>
```

```
          See RFC 3031 for the association between
          MPLS labels and IP addresses.
          &lt;/paragraph&gt;
        &lt;/reference&gt;
      &lt;/field&gt;

      &lt;field name="mplsTopLabelIPv6Address" dataType="ipv6Address"
             group="derived"
             dataTypeSemantics="identifier"
             elementId="140" applicability="data" status="current"&gt;
        &lt;description&gt;
          &lt;paragraph&gt;
            The IPv6 address of the system that the MPLS top label will
            cause this Flow to be forwarded to.
          &lt;/paragraph&gt;
        &lt;/description&gt;
        &lt;reference&gt;
          &lt;paragraph&gt;
          See RFC 3031 for the association between
          MPLS labels and IP addresses.
          &lt;/paragraph&gt;
        &lt;/reference&gt;
      &lt;/field&gt;

      &lt;field name="mplsVpnRouteDistinguisher" dataType="octetArray"
             group="derived"
             dataTypeSemantics="identifier"
             elementId="90" applicability="all" status="current"&gt;
        &lt;description&gt;
          &lt;paragraph&gt;
          The value of the VPN route distinguisher of a corresponding
          entry in a VPN routing and forwarding table.  Route
          distinguisher ensures that the same address can be used in
          several different MPLS VPNs and that it is possible for BGP to
          carry several completely different routes to that address, one
          for each VPN.  According to RFC 4364, the size of
          mplsVpnRouteDistinguisher is 8 octets.  However, in RFC 4382 an
          octet string with flexible length was chosen for representing a
          VPN route distinguisher by object MplsL3VpnRouteDistinguisher.
          This choice was made in order to be open to future changes of
          the size.  This idea was adopted when choosing octetArray as
          abstract data type for this Information Element.  The maximum
          length of this Information Element is 256 octets.
          &lt;/paragraph&gt;
        &lt;/description&gt;
        &lt;reference&gt;
          &lt;paragraph&gt;
          See RFC 4364 for the specification of the route
```

```
          distinguisher.  See RFC 4382 for the specification
          of the MPLS/BGP Layer 3 Virtual Private Network (VPN)
          Management Information Base.
          &lt;/paragraph&gt;
        &lt;/reference&gt;
      &lt;/field&gt;

      &lt;field name="minimumIpTotalLength" dataType="unsigned64"
             group="minMax"
             elementId="25" applicability="all" status="current"&gt;
        &lt;description&gt;
          &lt;paragraph&gt;
          Length of the smallest packet observed for this Flow.
          The packet length includes the IP header(s) length and
          the IP payload length.
          &lt;/paragraph&gt;
        &lt;/description&gt;
        &lt;reference&gt;
          &lt;paragraph&gt;
          See RFC 791 for the specification of the
          IPv4 total length.
          See RFC 2460 for the specification of the
          IPv6 payload length.
          See RFC 2675 for the specification of the
          IPv6 jumbo payload length.
          &lt;/paragraph&gt;
        &lt;/reference&gt;
        &lt;units&gt;octets&lt;/units&gt;
      &lt;/field&gt;

      &lt;field name="maximumIpTotalLength" dataType="unsigned64"
             group="minMax"
             elementId="26" applicability="all" status="current"&gt;
        &lt;description&gt;
          &lt;paragraph&gt;
          Length of the largest packet observed for this Flow.
          The packet length includes the IP header(s) length and
          the IP payload length.
          &lt;/paragraph&gt;
        &lt;/description&gt;
        &lt;reference&gt;
          &lt;paragraph&gt;
          See RFC 791 for the specification of the
          IPv4 total length.
          See RFC 2460 for the specification of the
          IPv6 payload length.
          See RFC 2675 for the specification of the
          IPv6 jumbo payload length.
```

```
        </paragraph>
      </reference>
      <units>octets</units>
    </field>

    <field name="minimumTTL" dataType="unsigned8"
           group="minMax"
           elementId="52" applicability="data" status="current">
      <description>
        <paragraph>
          Minimum TTL value observed for any packet in this Flow.
        </paragraph>
      </description>
      <reference>
        <paragraph>
        See RFC 791 for the definition of the IPv4
        Time to Live field.
        See RFC 2460 for the definition of the IPv6
        Hop Limit field.
        </paragraph>
      </reference>
      <units>hops</units>
    </field>

    <field name="maximumTTL" dataType="unsigned8"
           group="minMax"
           elementId="53" applicability="data" status="current">
      <description>
        <paragraph>
          Maximum TTL value observed for any packet in this Flow.
        </paragraph>
      </description>
      <reference>
       <paragraph>
       See RFC 791 for the definition of the IPv4
       Time to Live field.
       See RFC 2460 for the definition of the IPv6
       Hop Limit field.
       </paragraph>
      </reference>
      <units>hops</units>
    </field>

    <field name="ipv4Options" dataType="unsigned32"
           dataTypeSemantics="flags"
           group="minMax"
           elementId="208" applicability="all" status="current">
      <description>
```

<paragraph>
IPv4 options in packets of this Flow.
The information is encoded in a set of bit fields. For each valid IPv4 option type, there is a bit in this set. The bit is set to 1 if any observed packet of this Flow contains the corresponding IPv4 option type. Otherwise, if no observed packet of this Flow contained the respective IPv4 option type, the value of the corresponding bit is 0.
</paragraph>
<paragraph>
The list of valid IPv4 options is maintained by IANA. Note that for identifying an option not just the 5-bit Option Number, but all 8 bits of the Option Type need to match one of the IPv4 options specified at http://www.iana.org/assignments/ip-parameters.
</paragraph>
<paragraph>
Options are mapped to bits according to their option numbers. Option number X is mapped to bit X.
The mapping is illustrated by the figure below.
</paragraph>
<artwork>

```
        0      1      2      3      4      5      6      7
    +------+------+------+------+------+------+------+------+
    |      |  EXP |   to be assigned by IANA  |  QS  | UMP  | ...
    +------+------+------+------+------+------+------+------+


        8      9     10     11     12     13     14     15
    +------+------+------+------+------+------+------+------+
... | DPS  |NSAPA | SDB  |RTRALT|ADDEXT|  TR  | EIP  |IMITD | ...
    +------+------+------+------+------+------+------+------+


       16     17     18     19     20     21     22     23
    +------+------+------+------+------+------+------+------+
... |ENCODE| VISA | FINN | MTUR | MTUP | ZSU  | SSR  | SID  | ...
    +------+------+------+------+------+------+------+------+


       24     25     26     27     28     29     30     31
    +------+------+------+------+------+------+------+------+
... |  RR  |CIPSO |E-SEC |  TS  | LSR  | SEC  | NOP  | EOOL |
    +------+------+------+------+------+------+------+------+

      Type   Option
```

```
  Bit Value  Name    Reference
  ---+-----+-------+------------------------------------
   0     0   EOOL    End of Options List, RFC 791
   1     1   NOP     No Operation, RFC 791
   2   130   SEC     Security, RFC 1108
   3   131   LSR     Loose Source Route, RFC 791
   4    68   TS      Time Stamp, RFC 791
   5   133   E-SEC   Extended Security, RFC 1108
   6   134   CIPSO   Commercial Security
   7     7   RR      Record Route, RFC 791
   8   136   SID     Stream ID, RFC 791
   9   137   SSR     Strict Source Route, RFC 791
  10    10   ZSU     Experimental Measurement
  11    11   MTUP    (obsoleted) MTU Probe, RFC 1191
  12    12   MTUR    (obsoleted) MTU Reply, RFC 1191
  13   205   FINN    Experimental Flow Control
  14   142   VISA    Experimental Access Control
  15    15   ENCODE
  16   144   IMITD   IMI Traffic Descriptor
  17   145   EIP     Extended Internet Protocol, RFC 1385
  18    82   TR      Traceroute, RFC 3193
  19   147   ADDEXT  Address Extension
  20   148   RTRALT  Router Alert, RFC 2113
  21   149   SDB     Selective Directed Broadcast
  22   150   NSAPA   NSAP Address
  23   151   DPS     Dynamic Packet State
  24   152   UMP     Upstream Multicast Pkt.
  25    25   QS      Quick-Start
  30    30   EXP     RFC3692-style Experiment
  30    94   EXP     RFC3692-style Experiment
  30   158   EXP     RFC3692-style Experiment
  30   222   EXP     RFC3692-style Experiment
  ...  ...   ...     Further options numbers
                     may be assigned by IANA

    </artwork>
  </description>
  <reference>
    <paragraph>
    See RFC 791 for the definition of IPv4 options.
    See the list of IPv4 option numbers assigned by IANA
    at http://www.iana.org/assignments/ip-parameters.
    </paragraph>
  </reference>
</field>

<field name="ipv6ExtensionHeaders" dataType="unsigned32"
       dataTypeSemantics="flags"
```

```
            group="minMax"
            elementId="64" applicability="all" status="current">
       <description>
         <paragraph>
         IPv6 extension headers observed in packets of this Flow.
         The information is encoded in a set of bit fields.  For
         each IPv6 option header, there is a bit in this set.
         The bit is set to 1 if any observed packet of this Flow
         contains the corresponding IPv6 extension header.
         Otherwise, if no observed packet of this Flow contained
         the respective IPv6 extension header, the value of the
         corresponding bit is 0.
         </paragraph>
         <artwork>

               0     1     2     3     4     5     6     7
          +-----+-----+-----+-----+-----+-----+-----+-----+
          |                  Reserved                     | ...
          +-----+-----+-----+-----+-----+-----+-----+-----+

             8     9    10    11    12    13    14    15
          +-----+-----+-----+-----+-----+-----+-----+-----+
      ... |                  Reserved                     | ...
          +-----+-----+-----+-----+-----+-----+-----+-----+

            16    17    18    19    20    21    22    23
          +-----+-----+-----+-----+-----+-----+-----+-----+
      ... |         Reserved            | ESP | AH  | PAY | ...
          +-----+-----+-----+-----+-----+-----+-----+-----+

            24    25    26    27    28    29    30    31
          +-----+-----+-----+-----+-----+-----+-----+-----+
      ... | DST | HOP | Res | UNK | FRA0| RH  | FRA1| Res |
          +-----+-----+-----+-----+-----+-----+-----+-----+

        Bit    IPv6 Option   Description
       0, Res                Reserved
       1, FRA1     44        Fragmentation header - not first fragment
       2, RH       43        Routing header
       3, FRA0     44        Fragment header - first fragment
       4, UNK                Unknown Layer 4 header
                             (compressed, encrypted, not supported)
       5, Res                Reserved
       6, HOP       0        Hop-by-hop option header
       7, DST      60        Destination option header
       8, PAY     108        Payload compression header
       9, AH       51        Authentication Header
      10, ESP      50        Encrypted security payload
```

```
      11 to 31              Reserved
         </artwork>
       </description>
       <reference>
         <paragraph>
         See RFC 2460 for the general definition of
         IPv6 extension headers and for the specification of
         the hop-by-hop options header, the routing header,
         the fragment header, and the destination options header.
         See RFC 4302 for the specification of the
         authentication header.
         See RFC 4303 for the specification of the
         encapsulating security payload.
         </paragraph>
       </reference>
     </field>

     <field name="tcpControlBits" dataType="unsigned8"
            dataTypeSemantics="flags"
            group="minMax"
            elementId="6" applicability="all" status="current">
       <description>
         <paragraph>
         TCP control bits observed for packets of this Flow.
         The information is encoded in a set of bit fields.
         For each TCP control bit, there is a bit in this
         set.  A bit is set to 1 if any observed packet of this
         Flow has the corresponding TCP control bit set to 1.
         A value of 0 for a bit indicates that the corresponding
         bit was not set in any of the observed packets
         of this Flow.
         </paragraph>
         <artwork>
          0     1     2     3     4     5     6     7
      +-----+-----+-----+-----+-----+-----+-----+-----+
      |  Reserved | URG | ACK | PSH | RST | SYN | FIN |
      +-----+-----+-----+-----+-----+-----+-----+-----+

      Reserved:  Reserved for future use by TCP.  Must be zero.
           URG:  Urgent Pointer field significant
           ACK:  Acknowledgment field significant
           PSH:  Push Function
           RST:  Reset the connection
           SYN:  Synchronize sequence numbers
           FIN:  No more data from sender
         </artwork>
       </description>
       <reference>
```

```
         <paragraph>
         See RFC 793 for the definition of
         the TCP control bits in the TCP header.
         </paragraph>
       </reference>
     </field>

     <field name="tcpOptions" dataType="unsigned64"
            dataTypeSemantics="flags"
            group="minMax"
            elementId="209" applicability="all" status="current">
       <description>
         <paragraph>
        TCP options in packets of this Flow.
        The information is encoded in a set of bit fields.  For
        each TCP option, there is a bit in this set.
        The bit is set to 1 if any observed packet of this Flow
        contains the corresponding TCP option.
        Otherwise, if no observed packet of this Flow contained
        the respective TCP option, the value of the
        corresponding bit is 0.
         </paragraph>
         <paragraph>
         Options are mapped to bits according to their option
         numbers.  Option number X is mapped to bit X.
         TCP option numbers are maintained by IANA.
         </paragraph>
         <artwork>
             0     1     2     3     4     5     6     7
          +-----+-----+-----+-----+-----+-----+-----+-----+
          |  63 |  62 |  61 |  60 |  59 |  58 |  57 |  56 |  ...
          +-----+-----+-----+-----+-----+-----+-----+-----+

              8     9    10    11    12    13    14    15
          +-----+-----+-----+-----+-----+-----+-----+-----+
      ... |  55 |  54 |  53 |  52 |  51 |  50 |  49 |  48 |...
          +-----+-----+-----+-----+-----+-----+-----+-----+

             16    17    18    19    20    21    22    23
          +-----+-----+-----+-----+-----+-----+-----+-----+
      ... |  47 |  46 |  45 |  44 |  43 |  42 |  41 |  40 |...
          +-----+-----+-----+-----+-----+-----+-----+-----+

                                . . .

             56    57    58    59    60    61    62    63
          +-----+-----+-----+-----+-----+-----+-----+-----+
      ... |   7 |   6 |   5 |   4 |   3 |   2 |   1 |   0 |
```

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

         </artwork>
       </description>
       <reference>
         <paragraph>
         See RFC 793 for the definition of TCP options.
         See the list of TCP option numbers assigned by IANA
         at http://www.iana.org/assignments/tcp-parameters.
         </paragraph>
       </reference>
     </field>

     <field name="flowStartSeconds" dataType="dateTimeSeconds"
            group="timestamp"
            elementId="150" applicability="data" status="current">
       <description>
         <paragraph>
         The absolute timestamp of the first packet of this Flow.
         </paragraph>
       </description>
       <units>seconds</units>
     </field>

     <field name="flowEndSeconds" dataType="dateTimeSeconds"
            group="timestamp"
            elementId="151" applicability="data" status="current">
       <description>
         <paragraph>
         The absolute timestamp of the last packet of this Flow.
         </paragraph>
       </description>
       <units>seconds</units>
     </field>

     <field name="flowStartMilliseconds" dataType="dateTimeMilliseconds"
            group="timestamp"
            elementId="152" applicability="data" status="current">
       <description>
         <paragraph>
         The absolute timestamp of the first packet of this Flow.
         </paragraph>
       </description>
       <units>milliseconds</units>
     </field>

     <field name="flowEndMilliseconds" dataType="dateTimeMilliseconds"
```

```
             group="timestamp"
             elementId="153" applicability="data" status="current">
        <description>
          <paragraph>
          The absolute timestamp of the last packet of this Flow.
          </paragraph>
        </description>
        <units>milliseconds</units>
      </field>

      <field name="flowStartMicroseconds" dataType="dateTimeMicroseconds"
             group="timestamp"
             elementId="154" applicability="data" status="current">
        <description>
          <paragraph>
          The absolute timestamp of the first packet of this Flow.
          </paragraph>
        </description>
        <units>microseconds</units>
      </field>

      <field name="flowEndMicroseconds" dataType="dateTimeMicroseconds"
             group="timestamp"
             elementId="155" applicability="data" status="current">
        <description>
          <paragraph>
          The absolute timestamp of the last packet of this Flow.
          </paragraph>
        </description>
        <units>microseconds</units>
      </field>

      <field name="flowStartNanoseconds" dataType="dateTimeNanoseconds"
             group="timestamp"
             elementId="156" applicability="data" status="current">
        <description>
          <paragraph>
          The absolute timestamp of the first packet of this Flow.
          </paragraph>
        </description>
        <units>nanoseconds</units>
      </field>

      <field name="flowEndNanoseconds" dataType="dateTimeNanoseconds"
             group="timestamp"
             elementId="157" applicability="data" status="current">
        <description>
          <paragraph>
```

```
          The absolute timestamp of the last packet of this Flow.
          </paragraph>
        </description>
        <units>nanoseconds</units>
      </field>

      <field name="flowStartDeltaMicroseconds" dataType="unsigned32"
             group="timestamp"
             elementId="158" applicability="data" status="current">
        <description>
          <paragraph>
          This is a relative timestamp only valid within the scope
          of a single IPFIX Message.  It contains the negative time
          offset of the first observed packet of this Flow relative
          to the export time specified in the IPFIX Message Header.
          </paragraph>
        </description>
        <reference>
          <paragraph>
          See the IPFIX protocol specification [RFC5101] for the
          definition of the IPFIX Message Header.
          </paragraph>
        </reference>
        <units>microseconds</units>
      </field>

      <field name="flowEndDeltaMicroseconds" dataType="unsigned32"
             group="timestamp"
             elementId="159" applicability="data" status="current">
        <description>
          <paragraph>
          This is a relative timestamp only valid within the scope
          of a single IPFIX Message.  It contains the negative time
          offset of the last observed packet of this Flow relative
          to the export time specified in the IPFIX Message Header.
          </paragraph>
        </description>
        <reference>
          <paragraph>
          See the IPFIX protocol specification [RFC5101] for the
          definition of the IPFIX Message Header.
          </paragraph>
        </reference>
        <units>microseconds</units>
      </field>

      <field name="systemInitTimeMilliseconds"
             dataType="dateTimeMilliseconds"
```

```
           group="timestamp"
           elementId="160" applicability="data" status="current">
      &lt;description&gt;
        &lt;paragraph&gt;
        The absolute timestamp of the last (re-)initialization of the
        IPFIX Device.
        &lt;/paragraph&gt;
      &lt;/description&gt;
      &lt;units&gt;milliseconds&lt;/units&gt;
    &lt;/field&gt;

    &lt;field name="flowStartSysUpTime" dataType="unsigned32"
           group="timestamp"
           elementId="22" applicability="data" status="current">
      &lt;description&gt;
        &lt;paragraph&gt;
        The relative timestamp of the first packet of this Flow.
        It indicates the number of milliseconds since the
        last (re-)initialization of the IPFIX Device (sysUpTime).
        &lt;/paragraph&gt;
      &lt;/description&gt;
      &lt;units&gt;milliseconds&lt;/units&gt;
    &lt;/field&gt;

    &lt;field name="flowEndSysUpTime" dataType="unsigned32"
           group="timestamp"
           elementId="21" applicability="data" status="current">
      &lt;description&gt;
        &lt;paragraph&gt;
        The relative timestamp of the last packet of this Flow.
        It indicates the number of milliseconds since the
        last (re-)initialization of the IPFIX Device (sysUpTime).
        &lt;/paragraph&gt;
      &lt;/description&gt;
      &lt;units&gt;milliseconds&lt;/units&gt;
    &lt;/field&gt;

    &lt;field name="octetDeltaCount" dataType="unsigned64"
           dataTypeSemantics="deltaCounter"
           group="flowCounter"
           elementId="1" applicability="data" status="current">
      &lt;description&gt;
        &lt;paragraph&gt;
        The number of octets since the previous report (if any)
        in incoming packets for this Flow at the Observation Point.
        The number of octets includes IP header(s) and IP payload.
        &lt;/paragraph&gt;
      &lt;/description&gt;
```

```
      <units>octets</units>
    </field>

    <field name="postOctetDeltaCount" dataType="unsigned64"
           dataTypeSemantics="deltaCounter"
           group="flowCounter"
           elementId="23" applicability="data" status="current">
      <description>
        <paragraph>
        The definition of this Information Element is identical
        to the definition of Information Element
        'octetDeltaCount', except that it reports a
        potentially modified value caused by a middlebox
        function after the packet passed the Observation Point.
        </paragraph>
      </description>
      <units>octets</units>
    </field>

    <field name="octetDeltaSumOfSquares" dataType="unsigned64"
           group="flowCounter"
           elementId="198" applicability="data" status="current">
      <description>
        <paragraph>
        The sum of the squared numbers of octets per incoming
        packet since the previous report (if any) for this
        Flow at the Observation Point.
        The number of octets includes IP header(s) and IP payload.
        </paragraph>
      </description>
    </field>

    <field name="octetTotalCount" dataType="unsigned64"
           dataTypeSemantics="totalCounter"
           group="flowCounter"
           elementId="85" applicability="all" status="current">
      <description>
        <paragraph>
        The total number of octets in incoming packets
        for this Flow at the Observation Point since the Metering
        Process (re-)initialization for this Observation Point.  The
        number of octets includes IP header(s) and IP payload.
        </paragraph>
      </description>
      <units>octets</units>
    </field>

    <field name="postOctetTotalCount" dataType="unsigned64"
```

```
            dataTypeSemantics="totalCounter"
            group="flowCounter"
            elementId="171" applicability="all" status="current">
       <description>
         <paragraph>
         The definition of this Information Element is identical
         to the definition of Information Element
         'octetTotalCount', except that it reports a
         potentially modified value caused by a middlebox
         function after the packet passed the Observation Point.
         </paragraph>
       </description>
       <units>octets</units>
     </field>

     <field name="octetTotalSumOfSquares" dataType="unsigned64"
            group="flowCounter"
            elementId="199" applicability="all" status="current">
       <description>
         <paragraph>
         The total sum of the squared numbers of octets in incoming
         packets for this Flow at the Observation Point since the
         Metering Process (re-)initialization for this Observation
         Point.  The number of octets includes IP header(s) and IP
         payload.
         </paragraph>
       </description>
       <units>octets</units>
     </field>

     <field name="packetDeltaCount" dataType="unsigned64"
            dataTypeSemantics="deltaCounter"
            group="flowCounter"
            elementId="2" applicability="data" status="current">
       <description>
         <paragraph>
         The number of incoming packets since the previous report
         (if any) for this Flow at the Observation Point.
         </paragraph>
       </description>
       <units>packets</units>
     </field>

     <field name="postPacketDeltaCount" dataType="unsigned64"
            dataTypeSemantics="deltaCounter"
            group="flowCounter"
            elementId="24" applicability="data" status="current">
       <description>
```

```
        <paragraph>
        The definition of this Information Element is identical
        to the definition of Information Element
        'packetDeltaCount', except that it reports a
        potentially modified value caused by a middlebox
        function after the packet passed the Observation Point.
        </paragraph>
      </description>
      <units>packets</units>
    </field>

    <field name="packetTotalCount" dataType="unsigned64"
           dataTypeSemantics="totalCounter"
           group="flowCounter"
           elementId="86" applicability="all" status="current">
      <description>
        <paragraph>
        The total number of incoming packets for this Flow
        at the Observation Point since the Metering Process
        (re-)initialization for this Observation Point.
        </paragraph>
      </description>
      <units>packets</units>
    </field>

    <field name="postPacketTotalCount" dataType="unsigned64"
           dataTypeSemantics="totalCounter"
           group="flowCounter"
           elementId="172" applicability="all" status="current">
      <description>
        <paragraph>
        The definition of this Information Element is identical
        to the definition of Information Element
        'packetTotalCount', except that it reports a
        potentially modified value caused by a middlebox
        function after the packet passed the Observation Point.
        </paragraph>
      </description>
      <units>packets</units>
    </field>

    <field name="droppedOctetDeltaCount" dataType="unsigned64"
           dataTypeSemantics="deltaCounter"
           group="flowCounter"
           elementId="132" applicability="data" status="current">
      <description>
        <paragraph>
        The number of octets since the previous report (if any)
```

```
         in packets of this Flow dropped by packet treatment.
         The number of octets includes IP header(s) and IP payload.
         </paragraph>
       </description>
       <units>octets</units>
     </field>

     <field name="droppedPacketDeltaCount" dataType="unsigned64"
            dataTypeSemantics="deltaCounter"
            group="flowCounter"
            elementId="133" applicability="data" status="current">
       <description>
         <paragraph>
         The number of packets since the previous report (if any)
         of this Flow dropped by packet treatment.
         </paragraph>
       </description>
       <units>packets</units>
     </field>

     <field name="droppedOctetTotalCount" dataType="unsigned64"
            dataTypeSemantics="totalCounter"
            group="flowCounter"
            elementId="134" applicability="data" status="current">
       <description>
         <paragraph>
         The total number of octets in packets of this Flow dropped
         by packet treatment since the Metering Process
         (re-)initialization for this Observation Point.
         The number of octets includes IP header(s) and IP payload.
         </paragraph>
       </description>
       <units>octets</units>
     </field>

     <field name="droppedPacketTotalCount" dataType="unsigned64"
            dataTypeSemantics="totalCounter"
            group="flowCounter"
            elementId="135" applicability="data" status="current">
       <description>
         <paragraph>
         The number of packets of this Flow dropped by packet
         treatment since the Metering Process
         (re-)initialization for this Observation Point.
         </paragraph>
       </description>
       <units>packets</units>
     </field>
```

```
      <field name="postMCastPacketDeltaCount" dataType="unsigned64"
             dataTypeSemantics="deltaCounter"
             group="flowCounter"
             elementId="19" applicability="data" status="current">
        <description>
          <paragraph>
          The number of outgoing multicast packets since the
          previous report (if any) sent for packets of this Flow
          by a multicast daemon within the Observation Domain.
          This property cannot necessarily be observed at the
          Observation Point, but may be retrieved by other means.
          </paragraph>
        </description>
        <units>packets</units>
      </field>

      <field name="postMCastOctetDeltaCount" dataType="unsigned64"
             dataTypeSemantics="deltaCounter"
             group="flowCounter"
             elementId="20" applicability="data" status="current">
        <description>
          <paragraph>
          The number of octets since the previous report (if any)
          in outgoing multicast packets sent for packets of this
          Flow by a multicast daemon within the Observation Domain.
          This property cannot necessarily be observed at the
          Observation Point, but may be retrieved by other means.
          The number of octets includes IP header(s) and IP payload.
          </paragraph>
        </description>
        <units>octets</units>
      </field>

      <field name="postMCastPacketTotalCount" dataType="unsigned64"
             dataTypeSemantics="totalCounter"
             group="flowCounter"
             elementId="174" applicability="data" status="current">
        <description>
          <paragraph>
          The total number of outgoing multicast packets sent for
          packets of this Flow by a multicast daemon within the
          Observation Domain since the Metering Process
          (re-)initialization.  This property cannot necessarily
          be observed at the Observation Point, but may be retrieved
          by other means.
          </paragraph>
        </description>
        <units>packets</units>
```

</field>

<field name="postMCastOctetTotalCount" dataType="unsigned64"
dataTypeSemantics="totalCounter"
group="flowCounter"
elementId="175" applicability="data" status="current">
<description>
<paragraph>
The total number of octets in outgoing multicast packets
sent for packets of this Flow by a multicast daemon in the
Observation Domain since the Metering Process
(re-)initialization. This property cannot necessarily be
observed at the Observation Point, but may be retrieved by
other means.
The number of octets includes IP header(s) and IP payload.
</paragraph>
</description>
<units>octets</units>
</field>

<field name="tcpSynTotalCount" dataType="unsigned64"
dataTypeSemantics="totalCounter"
group="flowCounter"
elementId="218" applicability="data" status="current">
<description>
<paragraph>
The total number of packets of this Flow with
TCP "Synchronize sequence numbers" (SYN) flag set.
</paragraph>
</description>
<reference>
<paragraph>
See RFC 793 for the definition of the TCP SYN flag.
</paragraph>
</reference>
<units>packets</units>
</field>

<field name="tcpFinTotalCount" dataType="unsigned64"
dataTypeSemantics="totalCounter"
group="flowCounter"
elementId="219" applicability="data" status="current">
<description>
<paragraph>
The total number of packets of this Flow with
TCP "No more data from sender" (FIN) flag set.
</paragraph>
</description>

```
    <reference>
      <paragraph>
      See RFC 793 for the definition of the TCP FIN flag.
      </paragraph>
    </reference>
    <units>packets</units>
  </field>

  <field name="tcpRstTotalCount" dataType="unsigned64"
         dataTypeSemantics="totalCounter"
         group="flowCounter"
         elementId="220" applicability="data" status="current">
    <description>
      <paragraph>
       The total number of packets of this Flow with
       TCP "Reset the connection" (RST) flag set.
      </paragraph>
    </description>
    <reference>
      <paragraph>
      See RFC 793 for the definition of the TCP RST flag.
      </paragraph>
    </reference>
    <units>packets</units>
  </field>

  <field name="tcpPshTotalCount" dataType="unsigned64"
         dataTypeSemantics="totalCounter"
         group="flowCounter"
         elementId="221" applicability="data" status="current">
    <description>
      <paragraph>
       The total number of packets of this Flow with
       TCP "Push Function" (PSH) flag set.
      </paragraph>
    </description>
    <reference>
      <paragraph>
      See RFC 793 for the definition of the TCP PSH flag.
      </paragraph>
    </reference>
    <units>packets</units>
  </field>

  <field name="tcpAckTotalCount" dataType="unsigned64"
         dataTypeSemantics="totalCounter"
         group="flowCounter"
         elementId="222" applicability="data" status="current">
```

```
        <description>
          <paragraph>
           The total number of packets of this Flow with
           TCP "Acknowledgment field significant" (ACK) flag set.
          </paragraph>
        </description>
        <reference>
          <paragraph>
          See RFC 793 for the definition of the TCP ACK flag.
          </paragraph>
        </reference>
        <units>packets</units>
      </field>

      <field name="tcpUrgTotalCount" dataType="unsigned64"
             dataTypeSemantics="totalCounter"
             group="flowCounter"
             elementId="223" applicability="data" status="current">
        <description>
          <paragraph>
           The total number of packets of this Flow with
           TCP "Urgent Pointer field significant" (URG) flag set.
          </paragraph>
        </description>
        <reference>
          <paragraph>
          See RFC 793 for the definition of the TCP URG flag.
          </paragraph>
        </reference>
        <units>packets</units>
      </field>

      <field name="flowActiveTimeout" dataType="unsigned16"
             group="misc"
             elementId="36" applicability="all" status="current">
        <description>
          <paragraph>
          The number of seconds after which an active Flow is timed out
          anyway, even if there is still a continuous flow of packets.
          </paragraph>
        </description>
        <units>seconds</units>
      </field>

      <field name="flowIdleTimeout" dataType="unsigned16"
             group="misc"
             elementId="37" applicability="all" status="current">
        <description>
```

```
        <paragraph>
         A Flow is considered to be timed out if no packets belonging
         to the Flow have been observed for the number of seconds
         specified by this field.
        </paragraph>
      </description>
      <units>seconds</units>
    </field>

    <field name="flowEndReason" dataType="unsigned8"
           group="misc"
           dataTypeSemantics="identifier"
           elementId="136" applicability="data" status="current">
      <description>
        <paragraph>
        The reason for Flow termination.  The range of values includes
        the following:
        </paragraph>
        <artwork>
     0x01: idle timeout
           The Flow was terminated because it was considered to be
           idle.
     0x02: active timeout
           The Flow was terminated for reporting purposes while it was
           still active, for example, after the maximum lifetime of
           unreported Flows was reached.
     0x03: end of Flow detected
           The Flow was terminated because the Metering Process
           detected signals indicating the end of the Flow,
           for example, the TCP FIN flag.
     0x04: forced end
           The Flow was terminated because of some external event,
           for example, a shutdown of the Metering Process initiated
           by a network management application.
     0x05: lack of resources
           The Flow was terminated because of lack of resources
           available to the Metering Process and/or the Exporting
           Process.
        </artwork>
      </description>
    </field>

    <field name="flowDurationMilliseconds" dataType="unsigned32"
           group="misc"
           elementId="161" applicability="data" status="current">
      <description>
        <paragraph>
        The difference in time between the first observed packet
```

```
          of this Flow and the last observed packet of this Flow.
          </paragraph>
        </description>
        <units>milliseconds</units>
      </field>

      <field name="flowDurationMicroseconds" dataType="unsigned32"
             group="misc"
             elementId="162" applicability="data" status="current">
        <description>
          <paragraph>
          The difference in time between the first observed packet
          of this Flow and the last observed packet of this Flow.
          </paragraph>
        </description>
        <units>microseconds</units>
      </field>

      <field name="flowDirection" dataType="unsigned8"
             dataTypeSemantics="identifier"
             group="misc"
             elementId="61" applicability="data" status="current">
        <description>
          <paragraph>
          The direction of the Flow observed at the Observation
          Point.  There are only two values defined.
          </paragraph>
          <artwork>
          0x00: ingress flow
          0x01: egress flow
          </artwork>
        </description>
      </field>

      <field name="paddingOctets" dataType="octetArray"
             group="padding"
             elementId="210" applicability="option" status="current">
        <description>
          <paragraph>
            The value of this Information Element is always a sequence of
            0x00 values.
          </paragraph>
        </description>
      </field>

    </fieldDefinitions>
```

Appendix B. XML Specification of Abstract Data Types

This appendix contains a machine-readable description of the abstract data types to be used for IPFIX Information Elements and a machine-readable description of the template used for defining IPFIX Information Elements. Note that this appendix is of informational nature, while the text in Sections 2 and 3 (generated from this appendix) is normative.

At the same time, this appendix is also an XML schema that was used for creating the XML specification of Information Elements in Appendix A. It may also be used for specifying further Information Elements in extensions of the IPFIX information model. This schema and its namespace are registered by IANA at http://www.iana.org/assignments/xml-registry/schema/ipfix.xsd.

```
<?xml version="1.0" encoding="UTF-8"?>

<schema targetNamespace="urn:ietf:params:xml:ns:ipfix-info"
        xmlns:ipfix="urn:ietf:params:xml:ns:ipfix-info"
        xmlns="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified">

  <simpleType name="dataType">
    <restriction base="string">
      <enumeration value="unsigned8">
        <annotation>
          <documentation>The type "unsigned8" represents a
            non-negative integer value in the range of 0 to 255.
          </documentation>
        </annotation>
      </enumeration>

      <enumeration value="unsigned16">
        <annotation>
          <documentation>The type "unsigned16" represents a
            non-negative integer value in the range of 0 to 65535.
          </documentation>
        </annotation>
      </enumeration>

      <enumeration value="unsigned32">
        <annotation>
          <documentation>The type "unsigned32" represents a
             non-negative integer value in the range of 0 to
             4294967295.
          </documentation>
        </annotation>
      </enumeration>
```

```
         <enumeration value="unsigned64">
           <annotation>
             <documentation>The type "unsigned64" represents a
               non-negative integer value in the range of 0 to
               18446744073709551615.
             </documentation>
           </annotation>
         </enumeration>

         <enumeration value="signed8">
           <annotation>
             <documentation>The type "signed8" represents
               an integer value in the range of -128 to 127.
             </documentation>
           </annotation>
         </enumeration>

         <enumeration value="signed16">
           <annotation>
             <documentation>The type "signed16" represents an
               integer value in the range of -32768 to 32767.
             </documentation>
           </annotation>
         </enumeration>

         <enumeration value="signed32">
           <annotation>
             <documentation>The type "signed32" represents an
                integer value in the range of -2147483648 to
                2147483647.
             </documentation>
           </annotation>
         </enumeration>

         <enumeration value="signed64">
           <annotation>
             <documentation>The type "signed64" represents an
                integer value in the range of -9223372036854775808
                to 9223372036854775807.
             </documentation>
           </annotation>
         </enumeration>

         <enumeration value="float32">
           <annotation>
             <documentation>The type "float32" corresponds to an IEEE
               single-precision 32-bit floating point type as defined
               in [IEEE.754.1985].
```

```
            </documentation>
          </annotation>
        </enumeration>

        <enumeration value="float64">
          <annotation>
            <documentation>The type "float64" corresponds to an IEEE
              double-precision 64-bit floating point type as defined
              in [IEEE.754.1985].
            </documentation>
          </annotation>
        </enumeration>

        <enumeration value="boolean">
          <annotation>
            <documentation>The type "boolean" represents a binary
              value.  The only allowed values are "true" and "false".
            </documentation>
          </annotation>
        </enumeration>

        <enumeration value="macAddress">
          <annotation>
            <documentation>The type "macAddress" represents a
              string of 6 octets.
            </documentation>
          </annotation>
        </enumeration>

        <enumeration value="octetArray">
          <annotation>
            <documentation>The type "octetArray" represents a
           finite-length string of octets.
            </documentation>
          </annotation>
        </enumeration>

        <enumeration value="string">
          <annotation>
            <documentation>
              The type "string" represents a finite-length string
              of valid characters from the Unicode character encoding
              set [ISO.10646-1.1993].  Unicode allows for ASCII
              [ISO.646.1991] and many other international character
              sets to be used.
            </documentation>
          </annotation>
        </enumeration>
```

```
         <enumeration value="dateTimeSeconds">
           <annotation>
             <documentation>
               The type "dateTimeSeconds" represents a time value
               in units of seconds based on coordinated universal time
               (UTC).  The choice of an epoch, for example, 00:00 UTC,
               January 1, 1970, is left to corresponding encoding
               specifications for this type, for example, the IPFIX
               protocol specification.  Leap seconds are excluded.
               Note that transformation of values might be required
               between different encodings if different epoch values
               are used.
             </documentation>
           </annotation>
         </enumeration>

         <enumeration value="dateTimeMilliseconds">
           <annotation>
             <documentation>The type "dateTimeMilliseconds" represents
               a time value in units of milliseconds
               based on coordinated universal time (UTC).
               The choice of an epoch, for example,  00:00 UTC,
               January 1, 1970, is left to corresponding encoding
               specifications for this type, for example, the IPFIX
               protocol specification.  Leap seconds are excluded.
               Note that transformation of values might be required
               between different encodings if different epoch values
               are used.
             </documentation>
           </annotation>
         </enumeration>

         <enumeration value="dateTimeMicroseconds">
           <annotation>
             <documentation>The type "dateTimeMicroseconds" represents
               a time value in units of microseconds
               based on coordinated universal time (UTC).
               The choice of an epoch, for example,  00:00 UTC,
               January 1, 1970, is left to corresponding encoding
               specifications for this type, for example, the IPFIX
               protocol specification.  Leap seconds are excluded.
               Note that transformation of values might be required
               between different encodings if different epoch values
               are used.
             </documentation>
           </annotation>
         </enumeration>
```

```
        <enumeration value="dateTimeNanoseconds">
          <annotation>
            <documentation>The type "dateTimeNanoseconds" represents
              a time value in units of nanoseconds
              based on coordinated universal time (UTC).
              The choice of an epoch, for example,  00:00 UTC,
              January 1, 1970, is left to corresponding encoding
              specifications for this type, for example, the IPFIX
              protocol specification.  Leap seconds are excluded.
              Note that transformation of values might be required
              between different encodings if different epoch values
              are used.
            </documentation>
          </annotation>
        </enumeration>

        <enumeration value="ipv4Address">
          <annotation>
            <documentation>The type "ipv4Address" represents a value
              of an IPv4 address.
            </documentation>
          </annotation>
        </enumeration>
        <enumeration value="ipv6Address">
          <annotation>
            <documentation>The type "ipv6Address" represents a value
              of an IPv6 address.
            </documentation>
          </annotation>
        </enumeration>
      </restriction>
    </simpleType>

    <simpleType name="dataTypeSemantics">
      <restriction base="string">
        <enumeration value="quantity">
          <annotation>
            <documentation>
              A quantity value represents a discrete
              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.  If no semantic qualifier is
              given, the Information Elements that have an integral
              data type should behave as a quantity.
            </documentation>
          </annotation>
        </enumeration>
```

```
         <enumeration value="totalCounter">
           <annotation>
             <documentation>
               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
               RFC 2578 [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.
             </documentation>
           </annotation>
         </enumeration>

         <enumeration value="deltaCounter">
           <annotation>
             <documentation>
               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 its value is exported.
             </documentation>
           </annotation>
         </enumeration>

         <enumeration value="identifier">
           <annotation>
             <documentation>
               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.
```

```
            </documentation>
          </annotation>
        </enumeration>

        <enumeration value="flags">
          <annotation>
            <documentation>
              An integral value that actually represents a set of
              bit fields.  Logical operations are appropriate on
              such values, but not other mathematical operations.
              Flags should always be of an unsigned type.
            </documentation>
          </annotation>
        </enumeration>
      </restriction>
    </simpleType>

    <simpleType name="applicability">
      <restriction base="string">
        <enumeration value="data">
          <annotation>
            <documentation>
              Used for Information Elements that are applicable to
              Flow Records only.
            </documentation>
          </annotation>
        </enumeration>

        <enumeration value="option">
          <annotation>
            <documentation>
              Used for Information Elements that are applicable to
              option records only.
            </documentation>
          </annotation>
        </enumeration>

        <enumeration value="all">
          <annotation>
            <documentation>
              Used for Information Elements that are applicable to
              Flow Records as well as to option records.
            </documentation>
          </annotation>
        </enumeration>
      </restriction>
    </simpleType>
```

```
     <simpleType name="status">
       <restriction base="string">
         <enumeration value="current">
           <annotation>
             <documentation>
               Indicates that the Information Element definition
               is current and valid.
             </documentation>
           </annotation>
         </enumeration>

         <enumeration value="deprecated">
           <annotation>
             <documentation>
               Indicates that the Information Element definition is
               obsolete, but it permits new/continued implementation
               in order to foster interoperability with older/existing
               implementations.
             </documentation>
           </annotation>
         </enumeration>
         <enumeration value="obsolete">
           <annotation>
             <documentation>
               Indicates that the Information Element definition is
               obsolete and should not be implemented and/or can be
               removed if previously implemented.
             </documentation>
           </annotation>
         </enumeration>
       </restriction>
     </simpleType>

     <complexType name="text">
       <choice maxOccurs="unbounded" minOccurs="0">
         <element name="paragraph">
           <complexType mixed="true">
             <sequence>
                <element maxOccurs="unbounded" minOccurs="0"
                  name="xref">
                  <complexType>
                    <attribute name="target" type="string"
                      use="required"/>
                  </complexType>
                </element>
             </sequence>
           </complexType>
         </element>
```

```
        <element name="artwork">
          <simpleType>
            <restriction base="string"/>
          </simpleType>
        </element>
      </choice>
    </complexType>

    <simpleType name="range">
      <restriction base="string"/>
    </simpleType>
    <element name="fieldDefinitions">
      <complexType>
        <sequence>
          <element maxOccurs="unbounded" minOccurs="1" name="field">
            <complexType>
              <sequence>
                <element maxOccurs="1" minOccurs="1" name="description"
                         type="ipfix:text">
                  <annotation>
                    <documentation>
                      The semantics of this Information Element.
                      Describes how this Information Element is
                      derived from the Flow or other information
                      available to the observer.
                    </documentation>
                  </annotation>
                </element>

                <element maxOccurs="1" minOccurs="0" name="reference"
                         type="ipfix:text">
                  <annotation>
                    <documentation>
                      Identifies additional specifications that more
                      precisely define this item or provide additional
                      context for its use.
                    </documentation>
                  </annotation>
                </element>

                <element maxOccurs="1" minOccurs="0" name="units"
                         type="string">
                  <annotation>
                    <documentation>
                      If the Information Element is a measure of some
                      kind, the units identify what the measure is.
                    </documentation>
                  </annotation>
```

```
        </element>

        <element maxOccurs="1" minOccurs="0" name="range"
                 type="ipfix:range">
          <annotation>
            <documentation>
              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.
            </documentation>
          </annotation>
        </element>
      </sequence>

      <attribute name="name" type="string" use="required">
        <annotation>
          <documentation>
            A unique and meaningful name for the Information
            Element.
          </documentation>
        </annotation>
      </attribute>

      <attribute name="dataType" type="ipfix:dataType"
                 use="required">
        <annotation>
          <documentation>
            One of the types listed in Section 3.1 of this
            document or in a future extension of the
            information model.  The type space for attributes
            is constrained to facilitate implementation.  The
            existing type space does however encompass most
            basic types used in modern programming languages,
            as well as some derived types (such as ipv4Address)
            that are common to this domain and useful
            to distinguish.
          </documentation>
        </annotation>
      </attribute>

      <attribute name="dataTypeSemantics"
                 type="ipfix:dataTypeSemantics" use="optional">
        <annotation>
          <documentation>
            The integral types may be qualified by additional
            semantic details.  Valid values for the data type
```

```
          semantics are specified in Section 3.2 of this
          document or in a future extension of the
          information model.
        </documentation>
      </annotation>
    </attribute>

    <attribute name="elementId" type="nonNegativeInteger"
               use="required">
      <annotation>
        <documentation>
          A numeric identifier of the Information Element.
          If this identifier is used without an enterprise
          identifier (see [RFC5101] 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.
        </documentation>
      </annotation>
    </attribute>

    <attribute name="enterpriseId" type="nonNegativeInteger"
               use="optional">
      <annotation>
        <documentation>
          Enterprises may wish to define Information Elements
          without registering them with IANA, for example,
          for enterprise-internal purposes.  For such
          Information Elements, the Information Element
          identifier described above is not sufficient when
          the Information Element is used outside the
          enterprise.  If specifications of
          enterprise-specific Information Elements are made
          public and/or if enterprise-specific identifiers
          are used by the IPFIX protocol outside the
          enterprise, then the enterprise-specific
          identifier MUST be made globally unique by
          combining it with an enterprise identifier.
          Valid values for the enterpriseId are
          defined by IANA as Structure of Management
          Information (SMI) network management private
          enterprise codes.  They are defined at
          http://www.iana.org/assignments/enterprise-numbers.
        </documentation>
      </annotation>
    </attribute>
```

```
              <attribute name="applicability"
                         type="ipfix:applicability" use="optional">
                <annotation>
                  <documentation>
                    This property of an Information
                    Element indicates in which kind of records the
                    Information Element can be used.
                    Allowed values for this property are 'data',
                    'option', and 'all'.
                  </documentation>
                </annotation>
              </attribute>

              <attribute name="status" type="ipfix:status"
                         use="required">
                <annotation>
                  <documentation>
                    The status of the specification of this
                    Information Element.  Allowed values are 'current',
                    'deprecated', and 'obsolete'.
                  </documentation>
                </annotation>
              </attribute>
              <attribute name="group" type="string"
                         use="required">
                <annotation>
                  <documentation>to be done ...</documentation>
                </annotation>
              </attribute>

            </complexType>
          </element>
        </sequence>
      </complexType>

      <unique name="infoElementIdUnique">
        <selector xpath="field"/>

        <field xpath="elementId"/>
      </unique>
    </element>
  </schema>
```

Authors' 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

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

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

Paul Aitken
Cisco Systems, Inc.
96 Commercial Quay
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: January 7, 2012

B. Claise
Cisco Systems, Inc.
A. Kobayashi
NTT PF Lab.
B. Trammell
ETH Zurich
July 7, 2011

Specification of the Protocol for IPFIX Mediations
draft-claise-ipfix-mediation-protocol-04

Abstract

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

Status of this Memo

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

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

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

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-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 April, 2011.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents

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

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

The IPFIX architectural components in [RFC5470] consist of IPFIX Devices and IPFIX Collectors communicating using the IPFIX protocol [RFC5101], which specifies how to export IP Flow information. This protocol is designed to export information about IP traffic Flows and related measurement data, where a Flow is defined by a set of key attributes (e.g. source and destination IP address, source and destination port, etc.).

However, thanks to its Template mechanism, the IPFIX protocol can export any type of information, as long as the relevant Information Element is specified in the IPFIX Information Model [RFC5102], registered with IANA, or specified as an enterprise-specific Information Element. The specifications in the IPFIX protocol [RFC5101] have not been defined in the context of an IPFIX Mediator receiving, aggregating, correlating, anonymizing, etc... Flow Records from the one or multiple Exporters. Indeed, the IPFIX protocol must be adapted for Intermediate Processes, as defined in the IPFIX Mediation Reference Model as specified in the Figure A of [IPFIX-MED-FMWK], which is based on the IPFIX Mediation Problem Statement [RFC5982].

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

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

1.1. IPFIX Documents Overview

The IPFIX Protocol [RFC5101] provides network administrators with access to IP Flow information.

The architecture for the export of measured IP Flow information out of an IPFIX Exporting Process to a Collecting Process is defined in the IPFIX Architecture [RFC5470], per the requirements defined in RFC 3917 [RFC3917].

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

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

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

"IPFIX Mediation: Problem Statement" [RFC5982], describing the IPFIX Mediation applicability examples, along with some problems that network administrators have been facing, is the basis for the "IPFIX Mediation: Framework" [IPFIX-MED-FMWK]. This framework details the IPFIX Mediation reference model and the components of an IPFIX Mediator.

1.2. IPFIX Mediator Documents Overview

The "IPFIX Mediation: Problem Statement" [RFC5982] provides an overview of the applicability of Mediators, and defines requirements for Mediators in general terms. This document is of use largely to define the problems to be solved through the deployment of IPFIX Mediators, and to provide scope to the role of Mediators within an IPFIX collection infrastructure.

The "IPFIX Mediation: Framework" [IPFIX-MED-FMWK] provides more architectural details of the arrangement of Intermediate Processes within a Mediator.

The details of specific Intermediate Processes, when these have additional export specifications (e.g., metadata about the intermediate processing conveyed through IPFIX Options Templates), are each treated in their own document (e.g., the "IP Flow Anonymization Support" [RFC6235]). Documents specifying the operations of specific Intermediate Processes cover the operation of these Processes within the Mediator framework, and complying to the specifications given in this document; they may additionally specify the operation of the process independently, outside the context of a Mediator, when this is appropriate. As of today, these documents are:

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

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

1.3. Relationship with IPFIX and PSAMP

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

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

2. Terminology

The IPFIX-specific terms, such as Observation Domain, Flow, Flow Key, Metering Process, Exporting Process, Exporter, IPFIX Device, Collecting Process, Collector, Template, IPFIX Message, Message Header, Template Record, Data Record, Options Template Record, Set, Data Set, Information Element, and Transport Session, used in this document are defined in [RFC5101]. The PSAMP-specific terms used in this document, such as Filtering and Sampling are defined in [RFC5476].

The IPFIX Mediation terms related to the aggregation, such as the Interval, Aggregated Flow, and Aggregated Fonction are defined in [IPFIX-MED-AGGR].

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

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

Transport Session Information

The Transport Session is specified in [RFC5101]. In SCTP, the Transport Session Information is the SCTP association. In TCP and UDP, the Transport Session Information corresponds to a 5-tuple {Exporter IP address, Collector IP address, Exporter transport port, Collector transport port, transport protocol}.

Original Exporter

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

Original Observation Point

An Observation Point of the Original Exporter(s). In the case of the Intermediate Aggregation Process on an IPFIX Mediator, the Original Observation Point can be composed of a (set of) specific exporter(s), a (set of) specific interface(s) on an Exporter, a (set of) line card(s) on an Exporter, or any combinations of these.

IPFIX Mediation

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

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

Intermediate Process

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

Specific Intermediate Processes are described below. However, this is not an exhaustive list.

Intermediate Conversion Process

An Intermediate Conversion Process is an Intermediate Process that transforms non-IPFIX into IPFIX, or manages the relation among Templates and states of incoming/outgoing Transport Sessions (or equivalent for non IPFIX protocols) in the case of transport protocol conversion (e.g., from UDP to SCTP).

Intermediate Aggregation Process

An Intermediate Aggregation Process is an Intermediate Process that aggregates records based upon a set of Flow Keys or functions applied to fields from the record (e.g., binning and subnet aggregation).

Intermediate Correlation Process

An Intermediate Correlation Process is an Intermediate Process that adds information to records, noting correlations among them, or generates new records with correlated data from multiple records (e.g., the production of bidirectional flow records from unidirectional flow records).

Intermediate Selection Process

An Intermediate Selection Process is an Intermediate Process that selects records from a sequence based upon criteria-evaluated record values and passes only those records that match the criteria (e.g., Filtering only records from a given network to a given Collector).

Intermediate Anonymization Process

An Intermediate Anonymization Process is an Intermediate Process that transforms records in order to anonymize them, to protect the identity of the entities described by the records (e.g., by applying prefix-preserving pseudonymization of IP addresses).

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.

Template Mapping

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

Anonymization Record

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

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.

3. Specifications

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

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

- For the Intermediate Aggregation Process, the specifications in [IPFIX-MED-AGGR] MUST be followed.
- For the Intermediate Selection Process, the specifications in [IPFIX-MED-FLOWSEL] MUST be followed.
- For the Intermediate Anonymization Process, the specifications in [RFC6235] should be considered as guidelines as [RFC6235] is an experimental RFC.

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

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

3.1. Encoding of IPFIX Message Header

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

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Version Number          |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Export Time                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Sequence Number                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Observation Domain ID                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure A: IPFIX Message Header format

Message Header Field Descriptions

Version

Version of Flow Record format exported in this message. The value of this field is 0x000a for the current version, incrementing by one the version used in the NetFlow services export version 9 [RFC3954].

Length

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

Export Time

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

Sequence Number

Incremental sequence counter modulo 2^32 of all IPFIX Data Records sent on this PR-SCTP stream from the current Observation Domain by the Exporting Process. Check the specific meaning of this field in the sub-sections of section 10 when UDP or TCP is selected as the transport protocol. This value SHOULD be used by the Collecting Process to identify whether any IPFIX Data Records have been missed. Template and Options Template Records do not increase the Sequence Number.

Observation Domain ID

A 32-bit identifier of the Observation Domain that is locally unique to the Exporting Process. The Exporting Process uses the Observation Domain ID to uniquely identify to the Collecting Process the Observation Domain that metered the Flows. It is RECOMMENDED that this identifier is also unique per IPFIX Device. Collecting Processes SHOULD use the Transport Session and the Observation Domain ID field to separate different export streams originating from the same Exporting Process. The Observation Domain ID SHOULD be 0 when no specific Observation Domain ID is relevant for the entire IPFIX Message. For example, when exporting the Exporting Process Statistics, or in case of hierarchy of Collector when aggregated Data Records are exported.
Note: the Observation Domain Management is discussed in section 3.4.1.

3.2. Template Management

3.2.1. Template Management Without Template Records Change

The first case is a situation where the IPFIX Mediator doesn’t modify the (Options) Template Record(s) content. A typical example is an Intermediate Selection Process acting as distributor, which collects Flow Records from one or multiple 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 [RFC5101], the Template IDs are unique per Exporter, per Transport Session, and per Observation Domain. As there is no guarantee that, for similar Template Records, the Template IDs received on the incoming Transport Session and exported to the outgoing Transport Session would be same, the IPFIX Mediator MUST maintain a Template Mapping composed of similar received and exported (Options) Template Records:

- for each received (Options) Template Record: Template Record Flow Keys and non Flow Keys, Template ID, Observation Domain Id, and Transport Session Information
- for each exported (Options) Template Record: Template Record Flow Keys and non Flow Keys, Template ID, Collector, Observation Domain Id, and Transport Session Information

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

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

```
                                 Templ. .---------.
                                 ID 256 |         |
                                  .---->|Collector|<==>Customer
                                  |     |#1       |    #A
                                  |     |         |
                               RD=100:1 '---------'
.---------.Templ.  .---------.    |
|         |Id      |         |----'     .---------.
```

```
   |         |258     |         | RD=100:2 |          |
   |IPFIX    |------->|IPFIX    |--------->|Collector|<==>Customer
   |Exporter |        |Mediator | Templ.   |#2        |    #B
   |#1       |        |         | ID 257   |          |
   |         |        |         |----.     '---------'
   '---------'        '---------'    |
                                  RD=100:3
                             Templ.|     .---------.
                             ID    |     |         |
                             257   '---->|Collector|<==>Customer
                                         |#3       |    #C
                                         |         |
                                         '---------'
```

Figure B: Intermediate Aggregation Process Example

Template Entry A:
 Incoming Transport Session Information (from Exporter#1):
   Source IP: <Exporter#1 export IP address>
   Destination IP: <IPFIX Mediator IP address>
   Protocol: SCTP
   Source Port: <source port>
   Destination Port: 4739 (IPFIX)
 Observation Domain Id: <Observation Domain ID>
 Template Id: 258
   Flow Keys: <series of Flow Keys>
   Non Flow Keys: <series of non Flow Keys>

Template Entry B:
 Outgoing Transport Session Information (to Collector#1):
   Source IP: <IPFIX Mediator IP address>
   Destination IP: <IPFIX Collector#1 IP address>
   Protocol: SCTP
   Source Port: <source port>
   Destination Port: 4739 (IPFIX)
 Observation Domain Id: <Observation Domain ID>
 Template Id: 256
   Flow Keys: <series of Flow Keys>
   Non Flow Keys: <series of non Flow Keys>

Template Entry C:
 Outgoing Transport Session Information (to Collector#2):
   Source IP: <IPFIX Mediator IP address>
   Destination IP: <IPFIX Collector#2 IP address>
   Protocol: SCTP
   Source Port: <source port>

```
      Destination Port: 4739 (IPFIX)
    Observation Domain Id: <Observation Domain ID>
    Template Id: 257
      Flow Keys: <series of Flow Keys>
      Non Flow Keys: <series of non Flow Keys>

   Template Entry D:
    Outgoing Transport Session Information (to Collector#3):
      Source IP: <IPFIX Mediator IP address>
      Destination IP: <IPFIX Collector#3 IP address>
      Protocol: SCTP
      Source Port: <source port>
      Destination Port: 4739 (IPFIX)
    Observation Domain Id: <Observation Domain ID>
      Template Id: 257
      Flow Keys: <series of Flow Keys>
      Non Flow Keys: <series of non Flow Keys>
```

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

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

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.

3.2.2. Template Management With New Template Records

The second case is a situation where the IPFIX Mediator generates new (Options) Template Records compared to the received ones.

In such a 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 [IPFIX-MED-AGGR], 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 interesting to keep the correlation between the received (Options) Template Records and exported Derived Options) Template Records in the Template Mapping.

Therefore, the IPFIX Mediator MAY maintain a Template Mapping composed of received (Options) Template Records and exported derived Options) Template Records:

- for each received (Options) Template Record: Template Record Flow Keys and non Flow Keys, Template ID, Observation Domain, and Transport Session Information
- for each exported derived Options) Template Record: Template Record Flow Keys and non Flow Keys, Template ID, Collector, Observation Domain, and Transport Session Information

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

The following two examples illustrate this.

First, consider an IPFIX Mediator hosting an Intermediate Aggregation Process that generates time-series traffic octet counts per source IP address (as in the example in section 8.1 of [IPFIX-MED-AGGR]). Here, the Intermediate Process accepts Flow Records fitting any Template, discards all Information Elements other than the sourceIPv[46]Address and octetDeltaCount, aggregates these across all original Exporters in a given regular time interval, and exports Flow Records according to a Template Record containing flowStartTimeMilliseconds, flowEndTimeMilliseconds, sourceIPv[46]Address, and octetDeltaCount.

In this case, no Template Mapping is necessary. New Templates and Template Withdrawals in the Transport Sessions from the Original Exporters are handled as they would be at any Collecting Process. Records according to Templates which do not contain at least a timestamp, sourceIPv[46]Address, and octetDeltaCount IE are simply discarded by the Collector.

Next, consider a more generic case of this Intermediate Aggregation Process, which creates time-series aggregates across all Original Exporters, imposing a time interval but keeping a subset of the incoming Flow Key received from the Original

Exporter. In this case, a Template Mapping is necessary, as there is a relationship between incoming and outgoing Templates.

```
 .--------. tid 256 (src)
 |IPFIX   |
 |Exporter|----+
 |#1      |    |
 '--------'    |
 .--------.    |          .----------.               .---------.
 |IPFIX   |    '--------->|          |               |         |
 |Exporter|-------------->|IPFIX     |-------------->|IPFIX    |
 |#2      | tid 257 (src) |Mediator  |tid 256 (src)  |Collector|
 '--------'    +--------->|          |    257 (dst)  |         |
 .--------.    |          '----------'               '---------'
 |IPFIX   |    |
 |Exporter|----'
 |#3      | tid 257 (dst)
 '--------'
```

Figure C: Intermediate Aggregation Process Example

In Figure C, above, the Mediator accepts a Template Record containing only the sourceIPv4Address as the Flow Key from Exporters 1 and 2, and a Template Record containing only the destinationIPv4Address as the Flow Key from exporter 3. It exports time-series source aggregates as Template ID 256, and time-series destination aggregates as Template ID 257. The Template Entries in this case are as follows:

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: 256
    Flow Keys: sourceIPv4Address
    Non Flow Keys: octetDeltaCount, [others]

Template Entry B:
  Incoming Transport Session Information (from Exporter#2):
    Source IP: <Exporter#2 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: 257
      Flow Keys: sourceIPv4Address
      Non Flow Keys: octetDeltaCount, [others]

   Template Entry C:
    Incoming Transport Session Information (from Exporter#3):
      Source IP: <Exporter#3 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: 257
      Flow Keys: destinationIPv4Address
      Non Flow Keys: octetDeltaCount, [others]

   Template Entry D:
    Outgoing Transport Session Information (to IPFIX Collector):
      Source IP: <IPFIX Mediator export IP address>
      Destination IP: <IPFIX Collector IP address>
      Protocol: SCTP
      Source Port: <source port>
      Destination Port: 4739 (IPFIX)
    Observation Domain Id: <Observation Domain ID>
      Template Id: 256
      Flow Keys: sourceIPv4Address
      Non Flow Keys: octetDeltaCount

   Template Entry E:
    Outgoing Transport Session Information (to IPFIX Collector):
      Source IP: <IPFIX Mediator export IP address>
      Destination IP: <IPFIX Collector IP address>
      Protocol: SCTP
      Source Port: <source port>
      Destination Port: 4739 (IPFIX)
    Observation Domain Id: <Observation Domain ID>
      Template Id: 257
      Flow Keys: destinationIPv4Address
      Non Flow Keys: octetDeltaCount
```

The Template Mapping corresponding to the figure C can be displayed as:

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

      Template Entry C   <----> Template Entry E

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.

3.3. Time Management

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

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

The latter are difficult to manage for two reasons. First, they require constant translation, as the system initialization time of an intermediate system and the export time of an intermediate message will change across mediation operations. Further, relative timestamps introduce range problems. For example, when using the flowStartDeltaMicroseconds and flowEndDeltaMicroseconds Information Elements [RFC5102], the Data Record must be exported within a maximum of 71 minutes after its creation. Otherwise, the 32-bit counter would not be sufficient to contain the flow start time offset. Those time constraints might be incompatible with some of the Intermediate Processes: Intermediate Aggregation Process (temporal) and Intermediate Correlation Process, for example.

When an Intermediate Aggregation Process aggregates information from different Flow Records, the typical reporting times SHOULD BE the minimum of the start times and the maximum of the end times. However, if the Flow Records do not overlap, i.e. if there is a time gap between the times in the Flow Records, then the report may be inaccurate. The IPFIX Mediator is only reporting what it knows, on the basis of the information made available to it - and there may not have been any data to observe during the gap. Then again, if there is an overlap in timestamps, there’s the potential of double-accounting: different Observation Points may have observed the same traffic simultaneously. Therefore, as there is not a single rule that fits all different situations, the precise rules of applying the Flow Record timestamps in IPFIX Mediators is out of the scope of this document. However, some more specifications related to the specific case of aggregation in space and time are specified in [IPFIX-MED-AGGR], and MUST be followed.

3.4. Observation Point Management

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

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

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

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

Any combination of the above examples is possible. For example, in case of an Intermediate Aggregation Process, an Original Observation Point can 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
```

If the Original Observation Point is composed of a list, then the IPFIX Structured Data [IPFIX-STRUCT] 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 back the previous example, the following encoding can be used:

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

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

When an IPFIX Mediator 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 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, via the IPFIX Mediator(s) is out of the scope of this specification.

3.4.1. Observation Domain Management

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 3.1 above, and section 3.1 of [RFC5101].

IPFIX Mediators that do not change (Options) Template Records MUST maintain a Template Mapping, as detailed in section 3.2.1, 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 unchanged, as in section 3.2.2, 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.

3.5. Specific Reporting Requirements

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

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

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

3.5.1. The Flow Keys Options Template

Exactly like the IPFIX protocol [RFC5101], the Flow Keys Option Template specifies the structure of a Data Record for reporting the Flow Keys of reported Flows. A Flow Keys Data Record extends a particular Template Record that is referenced by its templateId identifier. The Template Record is extended by specifying which of the Information Elements contained in the corresponding Data Records describe Flow properties that serve as Flow Keys of the reported Flow.

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

templateId	An identifier of a Template. This Information Element MUST be defined as a Scope Field.
flowKeyIndicator	Bitmap with the positions of the Flow Keys in the Data Records.

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

3.5.2. IPFIX Protocol Options Template

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

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

3.5.3. IPFIX Mediator Options Template

There is no need for a specific Options Template for the IPFIX Mediator; instead, each Intermediate Process type requires some particular metadata. For example, a specification of IPFIX flow Anonymization including an Options Template for the export of metadata about Anonymized flows is described in [RFC6235]; when Anonymizing Flows Records, IPFIX Mediators SHOULD add the Options Template specified therein to annotate the exported data.

Transport Session Management SCTP [RFC4960] using the PR-SCTP extension specified in [RFC3758] MUST be implemented by all compliant IPFIX Mediator implementations. UDP [UDP] MAY also be implemented by compliant IPFIX Mediator implementations. TCP [TCP] MAY also be implemented by IPFIX Mediator compliant implementations.

PR-SCTP SHOULD be used in deployments where IPFIX Mediators and Collectors are communicating over links that are susceptible to congestion. PR-SCTP is capable of providing any required degree of reliability.

TCP MAY be used in deployments where IPFIX Mediators and Collectors communicate over links that are susceptible to congestion, but PR-SCTP is preferred due to its ability to limit back pressure on Exporters and its message versus stream orientation.

UDP MAY be used, although it is not a congestion-aware protocol. However, the IPFIX traffic between IPFIX Mediator and Collector MUST run in an environment where IPFIX traffic has been provisioned for, or is contained through some other means.

3.6. The Collecting Process's Side

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

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

3.7. Configuration Management

In some cases such as an Intermediate Aggregation Process aggregating Flow Records from multiple Original Exporters, a consistent configuration of the Metering Processes and Exporting Processes on these Original offers some advantages. For example, consistent active timeout, inactive timeout, and/or consistent export time allows to compare the number of the Flow Records per period of time. For example, consistent Sampling algorithm and parameters might allow to compare Flow Records accuracy.

While this is tempting to include all configuration parameters in Flow Records for the IPFIX Mediator to draw its own conclusion, the consistency of the configuration should be verified out of band, with the MIB modules ([RFC5815] and [PSAMP-MIB] or with the Configuration Data Model for IPFIX and PSAMP [IPFIX-CONF]

4. New Information Elements

EDITOR NOTE: please change the TBD1, TBD2, and TBD3, with the IANA newly assigned numbers.

4.1. - originalExporterIPv4Address

Description: The IPv4 address used by the Exporting Process on the Original Exporter. This is used by an IPFIX Mediator Exporting Process to identify the Original Exporter.

Abstract Data Type: ipv4Address

ElementId: TBD3

Status: Proposed

4.2. originalExporterIPv6Address

Description: The IPv6 address used by the Exporting Process on the Original Exporter. This is used by the IPFIX Mediator Exporting Process to identify the Original Exporter.

Abstract Data Type: ipv6Address

ElementId: TBD2

Status: Proposed

4.3. originalObservationDomainId

Description: An identifier of the Observation Domain on the Original Exporter, where the metered IP packets are observed. This is used by the IPFIX Mediator Exporting Process to identify an Observation Domain as received from the Original Exporter.

Abstract Data Type: unsigned32

ElementId: TBD3

Status: Proposed

5. Security Considerations

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

As they act as both IPFIX Collecting Processes and Exporting Processes, the Security Considerations for IPFIX [RFC5101] apply as well to Mediators. The Security Considerations for IPFIX Files [RFC5655] apply as well to IPFIX Mediators that write IPFIX Files or use them for internal storage. However, there are a few specific considerations that IPFIX Mediator implementations must take into account in addition.

By design, IPFIX Mediators are "men-in-the-middle": they intercede in the communication between an Original Exporter (or another upstream Mediator) and a downstream Collecting Process. This has two important implications for the level of confidentiality provided across an IPFIX Mediator, and the ability to protect data integrity and Original Exporter authenticity across a Mediator. These are addressed in more detail in the Security Considerations for Mediators in [IPFIX-MED-FMWK].

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

6. IANA Considerations

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

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

EDITOR’S NOTE: the XML specification in Appendix A must be updated with the elementID values allocated, i.e. TBD1, TBD2, and TDB3, must be replaced.

6.1. originalExporterIPv4Address

Name: originalExporterIPv4Address
Description:
The IPv4 address used by the Exporting Process on the Original Exporter. This is used by an IPFIX Mediator Exporting Process to identify the Original Exporter.
Abstract Data Type: ipv4Address
Data Type Semantics: identifier
ElementId: TBD1
Status: current

6.2. originalExporterIPv6Address

Name: originalExporterIPv6Address
Description:
The IPv6 address used by the Exporting Process on the Original Exporter. This is used by the IPFIX Mediator Exporting Process to identify the Original Exporter.
Abstract Data Type: ipv6Address
Data Type Semantics: identifier
ElementId: TBD2
Status: current

6.3. originalObservationDomainId

Name: originalObservationDomainId
Description:
An identifier of the Observation Domain on the Original Exporter, where the metered IP packets are observed. This is used by the IPFIX Mediator Exporting Process to identify an Observation Domain as received from the Original Exporter.
Abstract Data Type: unsigned32
Data Type Semantics: identifier
ElementId: TBD3
Status: current

7. References

7.1. Normative References

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

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

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

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

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

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

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

[IPFIX-MED-FLOWSEL] D'antonio, S., Zseby, T., Henke, C. and L. Peluso, "Flow Selection Techniques", draft-ietf-ipfix-flow-selection-tech-06.txt, Internet-Draft work in progress, May 2011.

[IPFIX-MED-AGGR] Trammell, B., Boschi, E., A. Wagner, and B. Claise, "Exporting Aggregated Flow Data using the IP Flow Information Export (IPFIX) Protocol", draft-trammell-ipfix-a9n-03.txt, Internet-Draft work in progress, June 2011.

[IPFIX-STRUCT] Claise, B., Dhandapani, G., Aitken, P., and S. Yates, "Export of Structured Data in IPFIX", draft-ietf-ipfix-structured-data-06.txt, Internet-Draft work in progress, May 2011.

[PSAMP-MIB] Dietz, T., Claise, B., and J. Quittek "Definitions of Managed Objects for Packet Sampling", draft-ietf-ipfix-psamp-mib-03.txt, Internet-Draft work in progress, March 2011.

[IPFIX-CONF] Muenz, G., Claise, B., and P. Aitken "Configuration Data Model for IPFIX and PSAMP", draft-ietf-ipfix-configuration-model-09, Internet-Draft work in progress, March 2011.

7.2. Informative References

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

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

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

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

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

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

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

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

[IPFIX-MED-FMWK] Kobayashi, A., Claise, B., Muenz, G., and K. Ishibashi, "IPFIX Mediation: Framework", RFC 6183, April 2011.

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

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

8. Author's Addresses

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

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

Atsushi Kobayashi
NTT Information Sharing Platform Laboratories

3-9-11 Midori-cho
Musashino-shi, Tokyo 180-8585
Japan

Phone: +81-422-59-3978
Email: akoba@nttv6.net
URI: http://www3.plala.or.jp/akoba/

Brian Trammell
ETH Zurich
Gloriastrasse 35
8092 Zurich
Switzerland

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

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

This appendix contains additions to the machine-readable description of the IPFIX information model coded in XML in Appendix A and Appendix B in [RFC5102]. Note that this appendix is of informational nature, while the text in Section 6. (generated from this appendix) is normative.

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


```
<field name="originalExporterIPv4Address"
         dataType="ipv4Address"
         group="config"
         elementId="TBD1" applicability="all" status="current">
    <description>
      <paragraph>
        The IPv4 address used by the Exporting Process on the
        Original Exporter. This is used by an IPFIX Mediator
        Exporting Process to identify the Original Exporter.
      </paragraph>
    </description>
  </field>


<field name="originalExporterIPv6Address"
         dataType="ipv6Address"
         group="config"
```

```
            elementId="TBD2" applicability="all" status="current">
       <description>
         <paragraph>
           The IPv6 address used by the Exporting Process on the
           Original Exporter. This is used by the IPFIX Mediator
           Exporting Process to identify the Original Exporter.
         </paragraph>
       </description>
     </field>

   <field name="originalObservationDomainId"
            dataType="unsigned32"
            group="config"
            elementId="TBD3" applicability="all" status="current">
       <description>
         <paragraph>
           An identifier of the Observation Domain on the Original
           Exporter, where the metered IP packets are observed.
           This is used by the IPFIX Mediator Exporting Process to
           identify an Observation Domain as received from the
           Original Exporter.
         </paragraph>
       </description>
     </field>
```

Network Working Group B. Claise, Ed.
Internet Draft Cisco Systems, Inc.
Obsoletes: 5101 October 26, 2011
Category: Standards Track
Expires: April 28, 2012

Specification of the IP Flow Information eXport (IPFIX) Protocol for the Exchange of IP Traffic Flow Information

draft-claise-ipfix-protocol-rfc5101bis-02

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on March 23, 2012.

Copyright Notice

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

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

Table of Contents

DONE:

Errata ID: 1655 (technical)
Errata ID: 2791 (technical)
Errata ID: 2814 (editorial)
Errata ID: 1818 (editorial)
Errata ID: 2792 (editorial)
Errata ID: 2888 (editorial)
Errata ID: 2889 (editorial)
Errata ID: 2890 (editorial)
Errata ID: 2891 (editorial)
Errata ID: 2892 (editorial)
Errata ID: 2903 (editorial)
Errata ID: 2761 (editorial)
Errata ID: 2762 (editorial)
Errata ID: 2763 (editorial)
Errata ID: 2764 (editorial)
Errata ID: 2852 (editorial)
Errata ID: 2857 (editorial)

Update all references to RFC5102bis and to RFC5815bis

Section 8: "a new sampling rate" has been removed from the list of examples that requires a new Template.

If the measurement parameters change such that a new Template is required, the Template MUST be withdrawn (using a Template Withdraw Message and a new Template definition) or an unused Template ID MUST be used. Examples of the measurement changes are: a new sampling rate, a new Flow expiration process, a new filtering definition, etc.

Updated the references

Updated the "Document overview" section.

Template and UDP: included the proposal at http://www.ietf.org/mail-archive/web/ipfix/current/msg06051.html

"time first flow dropped" and "time last flow dropped" inconsistency. See the discussion on the list.

Observation Domain Id: from "the same Exporting Process" to "the same Exporter". See http://www.ietf.org/mail-archive/web/ipfix/current/msg06078.html

Clarified the timestamps and updated the reference from RFC1305 to RFC5905

TO DO:

Resolution to the template lifetime mechanism for UDP

RFC2026 section 4.1.2: "The requirement for at least two independent and interoperable implementations applies to all of the options and features of the specification. In cases in which one or more options or features have not been demonstrated in at least two interoperable implementations, the specification may advance to the Draft Standard level only if those options or features are removed." The interop report from Prague is at http://www.ietf.org/proceedings/80/slides/ipfix-4.pdf Missing from this interop (and therefore, every interop):

1. DTLS over SCTP or UDP (5101 sec. 11.1)
2. ANY advanced template handling, withdrawal, stream separation, or reuse UDP template expiration (5101 sec. 10.3.6) template withdrawals (5101 sec. 8 para 8 et seq.)
3. SCTP export on any stream other than 0 (5101 sec 10.2.4.3)

1. Introduction

A data network with IP traffic primarily consists of IP flows passing through the network elements. It is often interesting, useful, or even required to have access to information about these flows that pass through the network elements for administrative or other purposes. The IPFIX 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 the protocol to achieve these aforementioned requirements. This document specifies in detail the representation of different flows, the additional data required for flow interpretation, packet format, transport mechanisms used, security concerns, etc.

1.1. IPFIX Documents Overview

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

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

IPFIX has a formal description of IPFIX Information Elements, their name, type and additional semantic information, as specified in [RFC5102bis], with the export of the Information Element types specified in [RFC5610].

[IPFIX-CONF] specifies a data model for configuring and monitoring IPFIX and PSAMP compliant devices using the NETCONF protocol, while the [RFC5815bis] specifies a MIB module for monitoring.

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

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

2. Terminology

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

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

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

Observation Point

An Observation Point is a location in the network where IP 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.

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

1. one or more packet header fields (e.g., destination IP address), transport header fields (e.g., destination port number), or application header fields (e.g., RTP header fields [RFC3550]).

2. one or more characteristics of the packet itself (e.g., number of MPLS labels, etc...).

3. one or more of fields derived from packet treatment (e.g., next hop IP address, the output interface, etc...).

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

This definition covers the range from a Flow containing all packets observed at a network interface to a Flow consisting of just a single packet between two applications. It includes packets selected by a sampling mechanism.

Flow Key

Each of the fields that:

1. belong to the packet header (e.g., destination IP address),

2. are a property of the packet itself (e.g., packet length),

3. are derived from packet treatment (e.g., Autonomous System (AS) number),

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

Flow Record

A Flow Record contains information about a specific Flow that was observed at an Observation Point. A Flow Record contains measured properties of the Flow (e.g., the total number of bytes for all the Flow's packets) and usually characteristic properties of the

Flow (e.g., source IP address).

Metering Process

The Metering Process generates Flow Records. Inputs to the process are packet headers and characteristics observed at an Observation Point, and packet treatment at the Observation Point (for example, the selected output interface).

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

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

Exporting Process

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

Exporter

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

IPFIX Device

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

Collecting Process

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

Collector

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

Template

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

IPFIX Message

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

Message Header

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

Template Record

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

Data Record

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

Options Template Record

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

Set

Set is a generic term for a collection of records that have a similar structure. In an IPFIX Message, one or more Sets follow the Message Header.

There are three different types of Sets: Template Set, Options Template Set, and Data Set.

Template Set

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

Options Template Set

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

Data Set

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

Information Element

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

Transport Session

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

2.1. Terminology Summary Table

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

Figure A: Terminology Summary Table

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

A Template Set contains only Template Record(s).

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

3. IPFIX Message Format

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

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

```
+----------------------------------------------------+
| Message Header                                     |
+----------------------------------------------------+
| Set                                                |
+----------------------------------------------------+
| Set                                                |
+----------------------------------------------------+
  ...
+----------------------------------------------------+
| Set                                                |
+----------------------------------------------------+
```

Figure B: IPFIX Message Format

The Exporter MUST code all binary integers of the Message Header and the different Sets in network-byte order (also known as the big-endian byte ordering).

Following are some examples of IPFIX Messages:

1. An IPFIX Message consisting of interleaved Template, Data, and Options Template Sets -- A newly created Template is exported as soon as possible. So, if there is already an IPFIX Message with a Data Set that is being prepared for export, the Template and Options Template Sets are interleaved with this information, subject to availability of space.

```
+--------+--------------------------------------------------------+
|        | +----------+ +---------+     +-----------+ +---------+ |
|Message | | Template | | Data    |     | Options   | | Data    | |
```

```
| Header | | Set       | | Set      | ... | Template  | | Set      | |
|        | |           | |          |     | Set       | |          | |
|        | +-----------+ +----------+     +-----------+ +----------+ |
+--------+------------------------------------------------------------+
```

Figure C: IPFIX Message, Example 1

2. An IPFIX Message consisting entirely of Data Sets -- After the appropriate Template Records have been defined and transmitted to the Collecting Process, the majority of IPFIX Messages consist solely of Data Sets.

```
+--------+----------------------------------------------+
|        | +----------+     +----------+      +----------+ |
|Message | | Data     |     | Data     |      | Data     | |
| Header | | Set      | ... | Set      | ...  | Set      | |
|        | +----------+     +----------+      +----------+ |
+--------+----------------------------------------------+
```

Figure D: IPFIX Message, Example 2

3. An IPFIX Message consisting entirely of Template and Options Template Sets.

```
+--------+-------------------------------------------------+
|        | +-----------+     +-----------+      +-----------+ |
|Message | | Template  |     | Template  |      | Options   | |
| Header | | Set       | ... | Set       | ...  | Template  | |
|        | |           |     |           |      | Set       | |
|        | +-----------+     +-----------+      +-----------+ |
+--------+-------------------------------------------------+
```

Figure E: IPFIX Message, Example 3

3.1. Message Header Format

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

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Version Number          |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Export Time                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Sequence Number                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Observation Domain ID                      |
```

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

Figure F: IPFIX Message Header Format

Message Header Field Descriptions:

Version

Version of Flow Record format exported in this message. The value of this field is 0x000a for the current version, incrementing by one the version used in the NetFlow services export version 9 [RFC3954].

Length

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

Export Time

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

Sequence Number

Incremental sequence counter modulo 2^32 of all IPFIX Data Records sent on this PR-SCTP stream from the current Observation Domain by the Exporting Process. Check the specific meaning of this field in the subsections of Section 10 when UDP or TCP is selected as the transport protocol. This value SHOULD be used by the Collecting Process to identify whether any IPFIX Data Records have been missed. Template and Options Template Records do not increase the Sequence Number.

Observation Domain ID

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

Collectors when aggregated Data Records are exported.

3.2. Field Specifier Format

Vendors need the ability to define proprietary Information Elements, because, for example, they are delivering a pre-standards product, or the Information Element is, in some way, commercially sensitive. This section describes the Field Specifier format for both IETF-specified Information Elements [RFC5102bis] and enterprise-specific Information Elements.

The Information Elements are identified by the Information Element identifier. When the Enterprise bit is set to 0, the corresponding Information Element identifier will report an IETF-specified Information Element, and the Enterprise Number MUST NOT be present. When the Enterprise bit is set to 1, the corresponding Information Element identifier will report an enterprise-specific Information Element; the Enterprise Number MUST be present. An example of this is shown in Section A.4.2.

The Field Specifier format is shown in Figure G.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|E|  Information Element ident. |        Field Length           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Enterprise Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure G: Field Specifier Format

Where:

E

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

Information Element identifier

A numeric value that represents the type of Information Element. Refer to [RFC5102bis].

Field Length

The length of the corresponding encoded Information Element, in octets. Refer to [RFC5102bis]. The field length may be smaller than the definition in [RFC5102bis] if the reduced size encoding is used (see Section 6.2). The value 65535 is reserved for variable-length Information Elements (see Section 7).

Enterprise Number

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

3.3. Set and Set Header Format

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

3.3.1. Set Format

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

```
+--------------------------------------------------+
| Set Header                                       |
+--------------------------------------------------+
| record                                           |
+--------------------------------------------------+
| record                                           |
+--------------------------------------------------+
 ...
+--------------------------------------------------+
| record                                           |
+--------------------------------------------------+
| Padding (opt.)                                   |
+--------------------------------------------------+
```

Figure H: Set Format

The Set Field Definitions are as follows:

Set Header

The Set Header Format is defined in Section 3.3.2.

Record

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

Padding

The Exporting Process MAY insert some padding octets, so that the subsequent Set starts at an aligned boundary. For security reasons, the padding octet(s) MUST be composed of zero (0) valued octets. The padding length MUST be shorter than any allowable record in this Set. If padding of the IPFIX Message is desired in combination with very short records, then the padding Information Element 'paddingOctets' [RFC5102bis] can be used for padding records such that their length is increased to a multiple of 4 or 8 octets. Because Template Sets are always 4-octet aligned by definition, padding is only needed in case of other alignments e.g., on 8-octet boundaries.

3.3.2. Set Header Format

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

```
 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               |          Length               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure I: Set Header Format

The Set Header Field Definitions are as follows:

Set ID

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

Length

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

3.4. Record Format

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

3.4.1. Template Record Format

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

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

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

Figure J: Template Record Format

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

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Template ID (> 255)      |         Field Count           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure K: Template Record Header Format

The Template Record Header Field Definitions are as follows:

Template ID

Each of the newly generated Template Records is given a unique Template ID. This uniqueness is local to the Transport Session and Observation Domain that generated the Template ID. Template IDs 0-255 are reserved for Template Sets, Options Template Sets, and other reserved Sets yet to be created. Template IDs of Data Sets are numbered from 256 to 65535. There are no constraints regarding the order of the Template ID allocation.

Field Count

Number of fields in this Template Record.

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

```
 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 are defined by the IETF (Enterprise bit = 0) and, therefore, do not need an Enterprise Number to identify them.

3.4.2. Options Template Record Format

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

One Options Template Record example is the "Flow Keys", which reports the Flow Keys for a Template, which is defined as the scope. Another example is the "Template configuration", which reports the configuration sampling parameter(s) for the Template, which is defined as the scope.

3.4.2.1. Scope

The scope, which is only available in the Options Template Set, gives the context of the reported Information Elements in the Data Records. Note that the IPFIX Message Header already contains the Observation Domain ID (the identifier of the Observation Domain). If not zero, this Observation Domain ID can be considered as an implicit scope for the Data Records in the IPFIX Message. The Observation Domain ID MUST be zero when the IPFIX Message contains Data Records with different Observation Domain ID values defined as scopes.

Multiple Scope Fields MAY be present in the Options Template Record, in which case, the composite scope is the combination of the scopes. For example, if the two scopes are defined as "metering process" and "template", the combined scope is this Template for this Metering Process. The order of the Scope Fields, as defined in the Options Template Record, is irrelevant in this case. However, if the order of the Scope Fields in the Options Template Record is relevant, the order of the Scope Fields MUST be used. For example, if the first scope defines the filtering function, while the second scope defines the sampling function, the order of the scope is important. Applying the sampling function first, followed by the filtering function, would lead to potentially different Data Records than applying the filtering function first, followed by the sampling function. In this case, the Collector deduces the function order by looking at the order of the scope in the Options Template Record.

The scope is an Information Element specified in the IPFIX Information Model [RFC5102bis]. An IPFIX-compliant implementation of the Collecting Process SHOULD support this minimum set of Information Elements as scope: LineCardId, TemplateId, exporterIPv4Address, exporterIPv6Address, and ingressInterface. Note that other Information Elements, such as meteringProcessId, exportingProcessId, observationDomainId, etc. are also valid scopes. The IPFIX protocol doesn’t prevent the use of any Information Elements for scope. However, some Information Element types don’t make sense if specified as scope; for example, the counter Information Elements.

Finally, note that the Scope Field Count MUST NOT be zero.

3.4.2.2. Options Template Record Format

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

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

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

Figure M: Options Template Record Format

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

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Template ID (> 255)   |         Field Count           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Scope Field Count        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure N: Options Template Record Header Format

The Options Template Record Header Field Definitions are as follows:

Template ID

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

Field Count

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

Scope Field Count

Number of scope fields in this Options Template Record. The Scope Fields are normal Fields except that they are interpreted as scope at the Collector. The Scope Field Count MUST NOT be zero.

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

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |          Set ID = 3           |          Length               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |         Template ID = 258     |         Field Count = N + M   |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |     Scope Field Count = N     |0|  Scope 1 Infor. Element Id. |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |     Scope 1 Field Length      |0|  Scope 2 Infor. Element Id. |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |     Scope 2 Field Length      |             ...               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |            ...                |1|  Scope N Infor. Element Id. |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |     Scope N Field Length      |   Scope N Enterprise Number ...
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
...  Scope N Enterprise Number   |1| Option 1 Infor. Element Id. |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |    Option 1 Field Length      |  Option 1 Enterprise Number ...
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
... Option 1 Enterprise Number   |              ...              |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |             ...               |0| Option M Infor. Element Id. |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |     Option M Field Length     |      Padding (optional)       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure O: 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".

```
+--------------------------------------------------+
| Field Value                                      |
+--------------------------------------------------+
| Field Value                                      |
+--------------------------------------------------+
 ...
+--------------------------------------------------+
| Field Value                                      |
+--------------------------------------------------+
```

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.

```
 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 = Template ID        |          Length               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Record 1 - Field Value 1    |   Record 1 - Field Value 2    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Record 1 - Field Value 3    |             ...               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Record 2 - Field Value 1    |   Record 2 - Field Value 2    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Record 2 - Field Value 3    |             ...               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Record 3 - Field Value 1    |   Record 3 - Field Value 2    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Record 3 - Field Value 3    |             ...               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              ...              |      Padding (optional)       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure Q: Data Set, Containing Data Records

4. Specific Reporting Requirements

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

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

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

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

4.1. The Metering Process Statistics Options Template

The Metering Process Statistics Options Template specifies the structure of a Data Record for reporting Metering Process statistics. It SHOULD contain the following Information Elements that are defined in [RFC5102bis]:

(scope) observationDomainId
An identifier of an Observation Domain that is locally unique to the Exporting Process. This Information Element MUST be defined as a Scope Field.

exportedMessageTotalCount
The total number of IPFIX Messages that the Exporting Process successfully sent to the Collecting Process since the Exporting Process re-initialization.

exportedFlowRecordTotalCount
The total number of Flow Records that the Exporting Process successfully sent to the Collecting Process since the Exporting Process re-initialization.

exportedOctetTotalCount
The total number of octets that the Exporting Process successfully sent to the Collecting Process since the Exporting Process re-initialization.

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

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

4.2. The Metering Process Reliability Statistics Options Template

The Metering Process Reliability Options Template specifies the structure of a Data Record for reporting lack of reliability in the Metering Process. It SHOULD contain the following Information Elements that are defined in [RFC5102bis]:

(scope) observationDomainId
An identifier of an Observation Domain that is locally unique to the Exporting Process. This Information Element MUST be defined as a Scope Field.

(scope) meteringProcessId
The identifier of the Metering Process for

which lack of reliability is reported. This Information Element MUST be defined as a Scope Field.

ignoredPacketTotalCount
: The total number of IP packets that the Metering Process did not process.

ignoredOctetTotalCount
: The total number of octets in observed IP packets that the Metering Process did not process.

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

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

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

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

Since the Metering Process Reliability Option Template will logically contain two identical timestamp Information Elements, and since the order of the Information Elements in the Template Records is not guaranteed, the Collecting Process MUST determine which is the oldest and the most recent timestamp in order the determine the right semantic behind the time first packet ignored and time last packet ignored Information Elements. Note that the counters wrap-up for the timestamps SHOULD also be taken into account.

4.3. The Exporting Process Reliability Statistics Options Template

The Exporting Process Reliability Options Template specifies the structure of a Data Record for reporting lack of reliability in the Exporting process. It SHOULD contain the following Information Elements that are defined in [RFC5102bis]:

(scope) Exporting Process ID
The identifier of the Exporting Process for which lack of reliability is reported. There are three Information Elements specified in [RFC5102bis] that can be used for this purpose: exporterIPv4Address, exporterIPv6Address, or

exportingProcessId. This Information Element MUST be defined as a Scope Field.

notSentFlowTotalCount
: The total number of Flows that were generated by the Metering Process and dropped by the Metering Process or by the Exporting Process instead of being sent to the Collecting Process.

notSentPacketTotalCount
: The total number of packets in Flow Records that were generated by the Metering Process and dropped by the Metering Process or by the Exporting Process instead of being sent to the Collecting Process.

notSentOctetTotalCount
: The total number of octets in packets in Flow Records that were generated by the Metering Process and dropped by the Metering Process or by the Exporting Process instead of being sent to the Collecting Process.

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

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

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

Since the Exporting Process Reliability Option Template will logically contain two identical timestamp Information Elements, and since the order of the Information Elements in the Template Records is not guaranteed, the Collecting Process MUST determine which is the oldest and the most recent timestamp in order the determine the right semantic behind the time first packet ignored and time last packet ignored Information Elements. Note that the counters wrap-up for the timestamps SHOULD also be taken into account.

4.4. The Flow Keys Options Template

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

The Flow Keys Options Template SHOULD contain the following Information Elements that are defined in [RFC5102bis]:

(scope) templateId	An identifier of a Template. This Information Element MUST be defined as a Scope Field.
flowKeyIndicator	Bitmap with the positions of the Flow Keys in the Data Records.

5. IPFIX Message Header Export Time and Flow Record Time

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

Certain time-related Information Elements may be expressed as an offset from this Export Time. For example, Data Records requiring a microsecond precision can export the flow start and end times with the flowStartMicroseconds and flowEndMicroseconds Information Elements [RFC5102bis], which encode the absolute time in microseconds in terms of the NTP epoch, 1 January 1900 at 00:00 UTC, in a 64-bit field. An alternate solution is to export the flowStartDeltaMicroseconds and flowEndDeltaMicroseconds Information Elements [RFC5102bis] in the Data Record, which respectively report the flow start and end time as negative offsets from the Export Time, as an unsigned 32-bit integer. This latter solution lowers the export bandwidth requirement, saving two bytes per timestamp, while increasing the load on the Exporter, as the Exporting Process must calculate the flowStartDeltaMicroseconds and flowEndDeltaMicroseconds of every single Data Record before exporting the IPFIX Message.

It must be noted that timestamps based on the Export Time impose some time constraints on the Data Records contained within the IPFIX Message. In the example of flowStartDeltaMicroseconds and flowEndDeltaMicroseconds Information Elements [RFC5102bis], the Data Record can only contain records with timestamps within 71 minutes of the Export Time. Otherwise, the 32-bit counter would not be sufficient to contain the flow start time offset.

6. Linkage with the Information Model

The Information Elements [RFC5102bis] MUST be sent in canonical format in network-byte order (also known as the big-endian byte ordering).

6.1. Encoding of IPFIX Data Types

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

6.1.1. Integral Data Types

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

6.1.2. Address Types

Address types -- macAddress, ipv4Address, and ipv6Address -- MUST be encoded the same way as the integral data types. The macAddress is treated as a 6-octet integer, the ipv4Address as a 4-octet integer, and the ipv6Address as a 16-octet integer.

6.1.3. float32

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

6.1.4. float64

The float64 data type MUST be encoded as an IEEE double-precision 64-bit floating point-type, as specified in [IEEE.754.1985].

6.1.5. boolean

The boolean data type is specified according to the TruthValue in [RFC2579]: it is an integer with the value 1 for true and a value 2 for false. Every other value is undefined. The boolean data type MUST be encoded in a single octet.

6.1.6. string and octetArray

The data type string represents a finite length string of valid characters of the Unicode character encoding set. The string data type MUST be encoded in UTF-8 format. The string is sent as an array of octets using an Information Element of fixed or variable length.

The length of the Information Element specifies the length of the octetArray.

6.1.7. dateTimeSeconds

The data type dateTimeSeconds is an unsigned 32 bit integer containing the number of seconds since the UNIX epoch, 1 January 1970 at 00:00 UTC. dateTimeSeconds is encoded identically to the IPFIX Message Header Export Time field. It can represent dates between 1 January 1970 and 8 February 2106.

6.1.8. dateTimeMilliseconds

The data type dateTimeMilliseconds is an unsigned 64-bit integer containing the number of milliseconds since the UNIX epoch, 1 January 1970 at 00:00 UTC. It can represent dates beginning on 1 January 1970 for approximately the next 500 billion years.

6.1.9 dateTimeMicroseconds

The data type dateTimeMicroseconds is a 64-bit field encoded according to the NTP Timestamp format as defined in section 6 of [RFC5905]. This field is made up of two unsigned 32-bit integers, Seconds and Fraction. The Seconds field is the number of seconds since the NTP epoch, 1 January 1900 at 00:00 UTC. The Fraction field is the fractional number of seconds in units of 1/(2^32) seconds (approximately 233 picoseconds). It can represent dates beginning between 1 January 1900 and 8 February 2036.

Note that dateTimeMicroseconds and dateTimeNanoseconds share an identical encoding. The dataTimeMicroseconds data type is intended only to represent timestamps of microsecond precision. Therefore, the bottom 11 bits of the fraction field MAY contain any value and MUST be ignored for all Information Elements of this data type (as 2^11 x 233 picoseconds = .477 microseconds).

6.1.10 dateTimeNanoseconds

The data type dateTimeNanoseconds is a 64-bit field encoded according to the NTP Timestamp format as defined in section 6 of [RFC5905]. This field is made up of two unsigned 32-bit integers, Seconds and Fraction. The Seconds field is the number of seconds since the NTP epoch, 1 January 1900 at 00:00 UTC. The Fraction field is the fractional number of seconds in units of 1/(2^32) seconds (approximately 233 picoseconds). It can represent dates beginning between 1 January 1900 and 8 February 2036.

Note that dateTimeMicroseconds and dateTimeNanoseconds share an identical encoding. There is no restriction on the interpretation of the Fraction field for the dateTimeNanoseconds data type.

6.2. Reduced Size Encoding of Integer and Float Types

Information Elements containing integer, string, float, and octetArray types in the information model MAY be encoded using fewer octets than those implied by their type in the information model definition [RFC5102bis], based on the assumption that the smaller size is sufficient to carry any value the Exporter may need to deliver. This reduces the network bandwidth requirement between the Exporter and the Collector. Note that the Information Element definitions [RFC5102bis] will always define the maximum encoding size.

For instance, the information model [RFC5102bis] defines byteCount as an unsigned64 type, which would require 64 bits. However, if the Exporter will never locally encounter the need to send a value larger than 4294967295, it may chose to send the value instead as an unsigned32. For example, a core router would require an unsigned64 byteCount, while an unsigned32 might be sufficient for an access router.

This behavior is indicated by the Exporter by specifying a type size with a smaller length than that associated with the assigned type of the Information Element. In the example above, the Exporter would place a length of 4 versus 8 in the Template.

If reduced size encoding is used, it MUST only 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 can also 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.

The reduced size encoding MUST NOT be applied to dateTimeMicroseconds or to dateTimeNanoseconds because these represent an inherent structure that would be destroyed by using less than the original number of bytes.

7. Variable-Length Information Element

The IPFIX Template mechanism is optimized for fixed-length Information Elements [RFC5102bis]. Where an Information Element has a variable length, the following mechanism MUST be used to carry the length information for both the IETF and proprietary Information Elements.

In the Template Set, the Information Element Field Length is recorded as 65535. This reserved length value notifies the Collecting Process that length of the Information Element will be carried in the Information Element content itself.

In most cases, the length of the Information Element will be less than 255 octets. The following length-encoding mechanism optimizes the overhead of carrying the Information Element length in this majority case. The length is carried in the octet before the Information Element, as shown in Figure R.

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | Length (< 255)|          Information Element                  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                      ... continuing as needed                 |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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.

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |      255      |      Length (0 to 65535)      |       IE      |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                      ... continuing as needed                 |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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 Template Management when using SCTP and PR-SCTP as the transport protocol. Any necessary changes to Template Management specifically related to TCP or UDP transport protocols are specified in Section 10.

The Exporting Process assigns and maintains the Template IDs per SCTP association for the Exporter’s Observation Domains. A newly created Template Record is assigned an unused Template ID by the Exporting Process.

If a specific Information Element is required by a Template, but is not available in observed packets, the Exporting Process MAY choose to export Flow Records without this Information Element in a Data Record defined by a new Template.

If an Information Element is required more than once in a Template, the different occurrences of this Information Element SHOULD follow the logical order of their treatments by the Metering Process. For example, if a selected packet goes through two hash functions, and if the two hash values are sent within a single Template, the first occurrence of the hash value should belong to the first hash function in the Metering Process. For example, when exporting the two source IP addresses of an IPv4 in IPv4 packets, the first sourceIPv4Address Information Element occurrence should be the IPv4 address of the outer header, while the second occurrence should be the inner header one.

Template Sets and Options Template Sets may be sent on any SCTP stream. Template Sets and Options Template Sets MUST be sent reliably, using SCTP-ordered delivery. As such, the Collecting Process MUST store the Template Record information for the duration of the SCTP association so that it can interpret the corresponding Data Records that are received in subsequent Data Sets.

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).

Different Observation Domains from the same SCTP association may use the same Template ID value to refer to different Templates.

The Templates that are not used anymore SHOULD be deleted. Before reusing a Template ID, the Template MUST be deleted. In order to delete an allocated Template, the Template is withdrawn through the use of a Template Withdrawal Message.

The Template Withdrawal Message MUST NOT be sent until sufficient time has elapsed to allow the Collecting Process to receive and process the last Data Record using this Template information. This time MUST be configurable. A suitable default value is 5 seconds after the last Data Record has been sent.

The Template ID from a withdrawn Template MUST NOT be reused until sufficient time has elapsed to allow for the Collecting Process to receive and process the Template Withdrawal Message.

A Template Withdrawal Message is a Template Record for that Template ID with a Field Count of 0. The format of the Template Withdrawal Message is shown in Figure T.

```
 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 or 3)       |          Length = 16          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Template ID N        |        Field Count = 0        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Template ID ...      |        Field Count = 0        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Template ID M        |        Field Count = 0        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure T: Template Withdrawal Message 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 Message, in that case, padding MAY be used.

The Template Withdrawal Message withdraws the Template IDs for the Observation Domain ID specified in the IPFIX Message Header.

The Template Withdrawal Message may be sent on any SCTP stream. The Template Withdrawal Message MUST be sent reliably, using SCTP-ordered delivery.

The Template Withdrawal Message MUST NOT contain new Template or Options Template Records.

If the measurement parameters change such that a new Template is required, the Template MUST be withdrawn (using a Template Withdrawal Message and a new Template definition) or an unused Template ID MUST be used. Examples of the measurement changes are: a new Flow expiration process, a new filtering definition, etc.

When the SCTP association shuts down or the Exporting Process restarts, all Template assignments are lost and Template IDs MUST be reassigned.

If the Metering Process restarts, the Exporting Process MUST either reuse the previously assigned Template ID for each Template, or it MUST withdraw the previously issued Template IDs by sending Template Withdrawal Message(s) before reusing them.

A Template Withdrawal Message to withdraw all Templates for the Observation Domain ID specified in the IPFIX Message Header MAY be used. Its format is shown in Figure U.

```
 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 Data Templates Withdrawal Message Format

A Template Withdrawal Message to withdraw all Options Templates for the Observation Domain ID specified in the IPFIX Message Header MAY be used. Its format is shown in Figure V.

```
 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 Message Format

When the SCTP association restarts, the Exporting Process MUST resend all the Template Records.

9. The Collecting Process’s Side

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

The Collecting Process SHOULD listen for a new association request from the Exporting Process. The Exporting Process will request a number of streams to use for export. An Exporting Process MAY request and support more than one stream per SCTP association.

If the Collecting Process receives a malformed IPFIX Message, it MUST reset the SCTP association, discard the IPFIX Message, and SHOULD log the error. Note that non-zero Set padding does not constitute a malformed IPFIX Message.

Template Sets and Options Template Sets are only sent once. The Collecting Process MUST store the Template Record information for the duration of the association so that it can interpret the corresponding Data Records that are received in subsequent Data Sets.

Template IDs are unique per SCTP association and per Observation Domain. If the Collecting Process receives a Template that has already been received but that has not previously been withdrawn (i.e., a Template Record from the same Exporter Observation Domain with the same Template ID received on the SCTP association), then the Collecting Process MUST shut down the association.

When an SCTP association is closed, the Collecting Process MUST discard all Templates received over that association and stop decoding IPFIX Messages that use those Templates.

The Collecting Process normally receives Template Records from the Exporting Process before receiving Data Records. The Data Records are then decoded and stored by the Collector. If the Template Records have not been received at the time Data Records are received, the Collecting Process MAY store the Data Records for a short period of time and decode them after the Template Records are received. 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.

The Collecting Process MUST note the Information Element identifier of any Information Element that it does not understand and MAY discard that Information Element from the Flow Record.

The Collector 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 may 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 a Collecting Process receives a Template Withdrawal Message, the Collecting Process MUST delete the corresponding Template Records associated with the specific SCTP association and specific Observation Domain, and stop decoding IPFIX Messages that use the withdrawn Templates.

If the Collecting Process receives a Template Withdrawal Message for a Template Record it has not received before on this SCTP association, it MUST reset the SCTP association, discard the IPFIX Message, and SHOULD log the error as it does for malformed IPFIX Messages.

A Collecting Process that receives IPFIX Messages from several Observation Domains on the same Transport Session MUST be aware that the uniqueness of the Template ID is not guaranteed across Observation Domains.

The Collector MUST support the use of Templates containing multiple occurrences of the similar Information Elements.

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

We need to differentiate between what must be implemented (so that operators can interoperably deploy compliant implementations from different vendors) and what should or could be used in various operational environments. We must also make sure that ALL implementations can operate in a congestion-aware and congestion-avoidance mode.

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.

PR-SCTP SHOULD be used in deployments where Exporters and Collectors are communicating over links that are susceptible to congestion. PR-SCTP is capable of providing any required degree of reliability.

TCP MAY be used in deployments where Exporters and Collectors communicate over links that are susceptible to congestion, but PR-SCTP is preferred due to its ability to limit back pressure on Exporters and its message versus stream orientation.

UDP MAY be used, although it is not a congestion-aware protocol. However, the IPFIX traffic between Exporter and Collector MUST run in an environment where IPFIX traffic has been provisioned for, or is contained through some other means.

10.2. SCTP

This section describes how IPFIX can be transported over SCTP [RFC4960] using the PR-SCTP [RFC3758] extension.

10.2.1. Congestion Avoidance

The SCTP transport protocol provides the required level of congestion avoidance by design.

SCTP will detect congestion in the end-to-end path between the IPFIX Exporting Process and the IPFIX Collecting Process, and limit the transfer rate accordingly. When an IPFIX Exporting Process has records to export, but detects that transmission by SCTP is temporarily impossible, it can either wait until sending is possible again, or it can decide to drop the record. In the latter case, the dropped export data MUST be accounted for, so that the amount of dropped export data can be reported.

10.2.2. Reliability

The SCTP transport protocol is by default reliable, but has the capability to deliver messages with partial reliability [RFC3758].

Using reliable SCTP messages for the IPFIX export is not in itself a guarantee that all Data Records will be delivered. If there is congestion on the link from the Exporting Process to the Collecting Process, or if a significant number of retransmissions are required, the send queues on the Exporting Process may fill up; the Exporting Process MAY either suspend, export, or discard the IPFIX Messages. If Data Records are discarded the IPFIX Sequence Numbers used for export MUST reflect the loss of data.

10.2.3. MTU

SCTP provides the required IPFIX Message fragmentation service based on path MTU discovery.

10.2.4. Exporting Process

10.2.4.1. Association Establishment

The IPFIX Exporting Process SHOULD initiate an SCTP association with the IPFIX Collecting Process. By default, the Collecting Process listens for connections on SCTP port 4739. By default, the Collecting Process listens for secure connections on SCTP port 4740 (refer to the Security Considerations section). By default, the Exporting Process tries to connect to one of these ports. It MUST be possible to configure both the Exporting and Collecting Processes to use a different SCTP port.

The Exporting Process MAY establish more than one association (connection "bundle" in SCTP terminology) to the Collecting Process.

An Exporting Process MAY support more than one active association to different Collecting Processes (including the case of different Collecting Processes on the same host).

10.2.4.2. Association Shutdown

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.

10.2.4.3. Stream

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.

10.2.4.4. Template Management

When the transport protocol is SCTP, the default Template Management described in Section 8 is used.

10.2.5. Collecting Process

When the transport protocol is SCTP, the default Collector processing described in Section 9 is used.

10.2.6. 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.

10.3. UDP

This section describes how IPFIX can be 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., over provisioned links compared to the maximum export rate from the Exporters.

10.3.2. Reliability

UDP is not a reliable transport protocol, and cannot guarantee delivery of messages. IPFIX Messages sent from the Exporting Process to the Collecting Process using UDP may therefore be lost. UDP MUST NOT be used unless the application can tolerate some loss of IPFIX Messages.

The Collecting Process SHOULD deduce the loss and reordering of IPFIX Data Records by looking at the discontinuities in the IPFIX Sequence Number. In the case of UDP, the IPFIX Sequence Number contains the total number of IPFIX Data Records sent for the UDP Transport Session prior to the receipt of this IPFIX Message, modulo 2^32. A Collector SHOULD detect out-of-sequence, dropped, or duplicate IPFIX Messages by tracking the Sequence Number. Templates sent from the Exporting Process to the Collecting Process using UDP as a transport MUST be re-sent at regular intervals, in case previous copies were lost.

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. Port Numbers

By default, the Collecting Process listens on the UDP port 4739. By default, the Collecting Process listens for secure connections on UDP port 4740 (refer to the "Security Considerations" section). By default, the Exporting Process tries to connect to one of these ports. It MUST be possible to configure both the Exporting and Collecting Processes to use a different UDP port.

10.3.5. Exporting Process

The Exporting Process MAY duplicate the IPFIX Message to the several Collecting Processes.

10.3.6. Template Management

When IPFIX uses UDP as the transport protocol, Template Sets and Options Template Sets MUST be re-sent at regular intervals. The frequency of the (Options) Template transmission MUST be configurable. The default value for the frequency of the (Options) Template transmission is 10 minutes. Note that the frequency of the (Options) Template transmission can be monitored and configured with the templateRefreshTimeout and optionsTemplateRefreshTimeout in [IPFIX-CONF]. 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.

In the event of configuration changes, the Exporting Process SHOULD send multiple copies of the new Template definitions, in different IPFIX Messages, at an accelerated rate. In such a case, it SHOULD transmit the changed Template Record(s) and Options Template Record(s), without any data, in advance to help ensure that the Collector will have the correct Template information before receiving the first data.

If the Options Template scope is defined in another Template, then both Templates SHOULD be sent in the same IPFIX Message. For example, if a Flow Key Options Template (see Section 4.4) is sent in an Options Template, then the associated Template SHOULD be sent in the same IPFIX Message.

Following a configuration change that can modify the interpretation of the Data Records (for example, a sampling rate change) a new Template ID MUST be used, and the old Template ID MUST NOT be reused until its lifetime (see Section 10.3.7) has expired.

If UDP is selected as the transport protocol, the Template Withdrawal Messages MUST NOT be used, as this method is inefficient due to the unreliable nature of UDP.

10.3.7. Collecting Process

The Collecting Process MUST associate a lifetime with each Template (or another definition of an identifier considered unique within the Transport Session) received via UDP. Templates (and similar definitions) not refreshed by the Exporting Process within the lifetime are expired at the Collecting Process. If the Template (or other definition) is not refreshed before that lifetime has expired, the Collecting Process MUST discard that definition and any current and future associated Data Records. In which case, an alarm MUST be logged. The Collecting Process MUST NOT decode any further Data Records that are associated with the expired Template. If a Template is refreshed with a Template Record that differs from the previously received Template Record, the Collecting Process SHOULD log a warning and replace the previously received Template Record with the new one. The Template lifetime at the Collecting Process MUST be at least 3 times higher than the Template refresh timeout configured on the Exporting Process.

Template IDs are unique per UDP session and per Observation Domain. At any given time, the Collecting Process SHOULD maintain the following for all the current Template Records and Options Template Records: <IPFIX Device, Exporter source UDP port, Observation Domain ID, Template ID, Template Definition, Last Received>.

The Collecting Process SHOULD accept Data Records without the associated Template Record (or other definitions) required to decode the Data Record. If the Template Records (or other definitions such as Common Properties) have not been received at the time Data Records are received, the Collecting Process SHOULD store the Data Records for a short period of time and decode them after the Template Records (or other definitions) are received. The short period of time MUST be lower than the lifetime of definitions associated with identifiers considered unique within the UDP session.

If the Collecting Process receives a malformed IPFIX Message, it MUST discard the IPFIX Message and SHOULD log the error.

10.3.8. Failover

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 can be transported over TCP [TCP].

10.4.1. Connection Management

10.4.1.1. Connection Establishment

The IPFIX Exporting Process initiates a TCP connection to the Collecting Process. By default, the Collecting Process listens for connections on TCP port 4739. By default, the Collecting Process listens for secure connections on TCP port 4740 (refer to the Security Considerations section). By default, the Exporting Process tries to connect to one of these ports. It MUST be possible to configure both the Exporting Process and the Collecting Process to use a different TCP port.

An Exporting Process MAY support more than one active connection to different Collecting Processes (including the case of different Collecting Processes on the same host).

The Exporter MAY log an alarm if the time to establish the connection exceeds a specified threshold, configurable on the Exporter.

10.4.1.2. Graceful Connection Release

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.

10.4.1.3. Restarting Interrupted Connections

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.

10.4.1.4. Failover

If the Collecting Process does not acknowledge the attempt by the Exporting Process to establish a connection, it will retry using the TCP exponential backoff feature.

If Collecting Process failover is supported by the Exporting Process, a second TCP connection MAY be opened in advance.

10.4.2. Data Transmission

Once a TCP connection is established, the Exporting Process starts sending IPFIX Messages to the Collecting Process.

10.4.2.1. IPFIX Message Encoding

IPFIX Messages are sent over the TCP connection without any special encoding. The Length field in the IPFIX Message Header defines the end of each IPFIX Message and thus the start of the next IPFIX Message. This means that IPFIX Messages cannot be interleaved.

In the case of TCP, the IPFIX Sequence Number contains the total number of IPFIX Data Records sent from this TCP connection, from the current Observation Domain by the Exporting Process, prior to the receipt of this IPFIX Message, modulo 2^32.

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 between different Observation Domains.

10.4.2.2. Template Management

For each Template, the Exporting Process MUST send the Template Record before exporting Data Records that refer to that Template.

Template IDs are unique per TCP connection and per Observation Domain. A Collecting Process MUST record all Template and Options Template Records for the duration of the connection, as an Exporting Process is not required to re-export Template Records.

When the TCP connection restarts, the Exporting Process MUST resend all the Template Records.

When a TCP connection is closed, the Collecting Process MUST discard all Templates received over that connection and stop decoding IPFIX Messages that use those Templates.

The Templates that are not used anymore SHOULD be deleted. Before reusing a Template ID, the Template MUST be deleted. In order to delete an allocated Template, the Template is withdrawn through the use of a Template Withdrawal Message over the TCP connection.

If the Collecting Process receives a malformed IPFIX Message, it MUST reset the TCP connection, discard the IPFIX Message, and SHOULD log the error.

10.4.2.3. Congestion Handling and Reliability

TCP ensures reliable delivery of data from the Exporting Process to the Collecting Process. TCP also controls the rate at which data can be sent from the Exporting Process to the Collecting Process, using a mechanism that takes into account both congestion in the network and the capabilities of the receiver.

Therefore, an IPFIX Exporting Process may not be able to send IPFIX Messages at the rate that the Metering Process generates it, either because of congestion in the network or because the Collecting Process cannot handle IPFIX Messages fast enough. As long as congestion is transient, the Exporting Process can buffer IPFIX

Messages for transmission. But such buffering is necessarily limited, both because of resource limitations and because of timeliness requirements, so ongoing and/or severe congestion may lead to a situation where the Exporting Process is blocked.

When an Exporting Process has Data Records to export but the transmission buffer is full, and it wants to avoid blocking, it can decide to drop some Data Records. The dropped Data Records MUST be accounted for, so that the amount can later be exported.

When an Exporting Process finds that the rate at which records should be exported is consistently higher than the rate at which TCP sending permits, it should provide back pressure to the Metering Processes. The Metering Process could then adapt by temporarily reducing the amount of data it generates, for example, using sampling or aggregation.

10.4.3. Collecting Process

The Collecting Process SHOULD listen for a new TCP connection from the Exporting Process.

If the Collecting Process receives a malformed IPFIX Message, it MUST reset the TCP connection, discard the IPFIX Message, and SHOULD log the error. Note that non-zero Set padding does not constitute a malformed IPFIX Message.

Template Sets and Options Template Sets are only sent once. The Collecting Process MUST store the Template Record information for the duration of the connection so that it can interpret the corresponding Data Records that are received in subsequent Data Sets.

Template IDs are unique per TCP connection and per Observation Domain. If the Collecting Process receives a Template that has already been received but that has not previously been withdrawn (i.e., a Template Record from the same Exporter Observation Domain with the same Template ID received on the TCP connection), then the Collecting Process MUST shut down the connection.

When a TCP connection is closed, the Collecting Process MUST discard all Templates received over that connection and stop decoding IPFIX Messages that use those Templates.

If a Collecting Process receives a Template Withdrawal Message, the Collecting Process MUST delete the corresponding Template Records associated with the specific TCP connection and specific Observation Domain, and stop decoding IPFIX Messages that use the withdrawn Templates.

If the Collecting Process receives a Template Withdrawal Message for a Template Record it has not received before on this TCP connection, it MUST reset the TCP association, discard the IPFIX Message, and SHOULD log the error as it does for malformed IPFIX Messages.

11. Security Considerations

The security considerations for the IPFIX protocol have been derived from an analysis of potential security threats, as discussed in the "Security Considerations" section of IPFIX requirements [RFC3917]. The requirements for IPFIX security are as follows:

1. IPFIX must provide a mechanism to ensure the confidentiality of IPFIX data transferred from an Exporting Process to a Collecting Process, in order to prevent disclosure of Flow Records transported via IPFIX.

2. IPFIX must provide a mechanism to ensure the integrity of IPFIX data transferred from an Exporting Process to a Collecting Process, in order to prevent the injection of incorrect data or control information (e.g., Templates) into an IPFIX Message stream.

3. IPFIX must provide a mechanism to authenticate IPFIX Collecting and Exporting Processes, to prevent the collection of data from an unauthorized Exporting Process or the export of data to an unauthorized Collecting Process.

Because IPFIX can be used to collect information for network forensics and billing purposes, attacks designed to confuse, disable, or take information from an IPFIX collection system may be seen as a prime objective during a sophisticated network attack.

An attacker in a position to inject false messages into an IPFIX Message stream can either affect the application using IPFIX (by falsifying data), or the IPFIX Collecting Process itself (by modifying or revoking Templates, or changing options); for this reason, IPFIX Message integrity is important.

The IPFIX Messages themselves may also contain information of value to an attacker, including information about the configuration of the network as well as end-user traffic and payload data, so care must be taken to confine their visibility to authorized users. When an Information Element containing end-user payload information is exported, it SHOULD be transmitted to the Collecting Process using a means that secures its contents against eavesdropping. Suitable mechanisms include the use of either a direct point-to-point connection or the use of an encryption mechanism. It is the responsibility of the Collecting Process to provide a satisfactory degree of security for this collected data, including, if necessary, anonymization of any reported data.

11.1. Applicability of TLS and DTLS

Transport Layer Security (TLS) [RFC5246] and Datagram Transport Layer Security (DTLS) [RFC4347] were designed to provide the confidentiality, integrity, and authentication assurances required by the IPFIX protocol, without the need for pre-shared keys.

With the mandatory SCTP and PR-SCTP transport protocols for IPFIX, DTLS [RFC4347] MUST be implemented. If UDP is selected as the IPFIX transport protocol, DTLS [RFC4347] MUST be implemented. If TCP is selected as the IPFIX transport protocol, TLS [RFC5246] MUST be implemented.

Note that DTLS is selected as the security mechanism for SCTP and PR-SCTP. Though TLS bindings to SCTP are defined in [RFC3436], they require all communication to be over reliable, bidirectional streams, and require one TLS connection per stream. This arrangement is not compatible with the rationale behind the choice of SCTP as an IPFIX transport protocol.

Note that using DTLS [RFC4347] has a vulnerability, i.e., a true man in the middle may attempt to take data out of an association and fool the sender into thinking that the data was actually received by the peer. In generic TLS for SCTP (and/or TCP), this is not possible. This means that the removal of a message may become hidden from the sender or receiver. Another vulnerability of using PR-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.

11.2. Usage

The IPFIX Exporting Process initiates the communication to the IPFIX Collecting Process, and acts as a TLS or DTLS client according to [RFC5246] and [RFC4347], while the IPFIX Collecting Process acts as a TLS or DTLS server. The DTLS client opens a secure connection on the SCTP port 4740 of the DTLS server if SCTP or PR-SCTP is selected as the transport protocol. The TLS client opens a secure connection on the TCP port 4740 of the TLS server if TCP is selected as the transport protocol. The DTLS client opens a secure connection on the UDP port 4740 of the DTLS server if UDP is selected as the transport protocol.

11.3. Authentication

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

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

The fully qualified domain name used to identify an IPFIX Collecting Process or Exporting Process may be stored either in a subjectAltName extension of type dNSName, or in the most specific Common Name field of the Subject field of the X.509 certificate. If both are present, the subjectAltName extension is given preference.

Internationalized domain names (IDN) in either the subjectAltName extension of type dNSName or the most specific Common Name field of the Subject field of the X.509 certificate MUST be encoded using Punycode [RFC3492] as described in [RFC5891], "Conversion Operations".

11.4. Protection against DoS Attacks

An attacker may mount a denial-of-service (DoS) attack against an IPFIX collection system either directly, by sending large amounts of traffic to a Collecting Process, or indirectly, by generating large amounts of traffic to be measured by a Metering Process.

Direct denial-of-service attacks can also involve state exhaustion, whether at the transport layer (e.g., by creating a large number of pending connections), or within the IPFIX Collecting Process itself (e.g., by sending Flow Records pending Template or scope information, a large amount of Options Template Records, etc.).

SCTP mandates a cookie-exchange mechanism designed to defend against SCTP state exhaustion denial-of-service attacks. Similarly, TCP provides the "SYN cookie" mechanism to mitigate state exhaustion; SYN cookies SHOULD be used by any Collecting Process accepting TCP connections. DTLS also provides cookie exchange to protect against DTLS server state exhaustion.

The reader should note that there is no way to prevent fake IPFIX Message processing (and state creation) for UDP & SCTP communication. The use of TLS and DTLS can obviously prevent the creation of fake states, but they are themselves prone to state exhaustion attacks. Therefore, Collector rate limiting SHOULD be used to protect TLS & DTLS (like limiting the number of new TLS or DTLS session per second to a sensible number).

IPFIX state exhaustion attacks can be mitigated by limiting the rate at which new connections or associations will be opened by the Collecting Process, the rate at which IPFIX Messages will be accepted by the Collecting Process, and adaptively limiting the amount of state kept, particularly records waiting on Templates. These rate and state limits MAY be provided by a Collecting Process; if provided, the limits SHOULD be user configurable.

Additionally, an IPFIX Collecting Process can eliminate the risk of state exhaustion attacks from untrusted nodes by requiring TLS or DTLS mutual authentication, causing the Collecting Process to accept IPFIX Messages only from trusted sources.

With respect to indirect denial of service, the behavior of IPFIX under overload conditions depends on the transport protocol in use. For IPFIX over TCP, TCP congestion control would cause the flow of IPFIX Messages to back off and eventually stall, blinding the IPFIX system. PR-SCTP improves upon this situation somewhat, as some IPFIX Messages would continue to be received by the Collecting Process due to the avoidance of head-of-line blocking by SCTP’s multiple streams and partial reliability features, possibly affording some visibility of the attack. The situation is similar with UDP, as some datagrams may continue to be received at the Collecting Process, effectively applying sampling to the IPFIX Message stream, implying that some forensics may be left.

To minimize IPFIX Message loss under overload conditions, some mechanism for service differentiation could be used to prioritize IPFIX traffic over other traffic on the same link. Alternatively, IPFIX Messages can be transported over a dedicated network. In this case, care must be taken to ensure that the dedicated network can handle the expected peak IPFIX Message traffic.

11.5. When DTLS or TLS Is Not an Option

The use of DTLS or TLS might not be possible in some cases due to performance issues or other operational concerns.

Without TLS or DTLS mutual authentication, IPFIX Exporting Processes and Collecting Processes can fall back on using IP source addresses to authenticate their peers. A policy of allocating Exporting Process and Collecting Process IP addresses from specified address ranges, and using ingress filtering to prevent spoofing, can improve the usefulness of this approach. Again, completely segregating IPFIX traffic on a dedicated network, where possible, can improve security even further. In any case, the use of open Collecting Processes (those that will accept IPFIX Messages from any Exporting Process regardless of IP address or identity) is discouraged.

Modern TCP and SCTP implementations are resistant to blind insertion attacks (see [RFC1948], [RFC4960]); however, UDP offers no such protection. For this reason, IPFIX Message traffic transported via UDP and not secured via DTLS SHOULD be protected via segregation to a dedicated network.

11.6. Logging an IPFIX Attack

IPFIX Collecting Processes MUST detect potential IPFIX Message insertion or loss conditions by tracking the IPFIX Sequence Number, and SHOULD provide a logging mechanism for reporting out-of-sequence messages. Note that an attacker may be able to exploit the handling of out-of-sequence messages at the Collecting Process, so care should be taken in handling these conditions. For example, a Collecting Process that simply resets the expected Sequence Number upon receipt of a later Sequence Number could be temporarily blinded by deliberate injection of later Sequence Numbers.

IPFIX Exporting and Collecting Processes SHOULD log any connection attempt that fails due to authentication failure, whether due to being presented an unauthorized or mismatched certificate during TLS or DTLS mutual authentication, or due to a connection attempt from an unauthorized IP address when TLS or DTLS is not in use.

IPFIX Exporting and Collecting Processes SHOULD detect and log any SCTP association reset or TCP connection reset.

11.7. Securing the Collector

The security of the Collector and its implementation is important to achieve overall security. However, it is outside the scope of this document.

12. IANA Considerations

IPFIX Messages use two fields with assigned values. These are the IPFIX Version Number, indicating which version of the IPFIX Protocol was used to export an IPFIX Message, and the IPFIX Set ID, indicating the type for each set of information within an IPFIX Message.

The IPFIX Version Number value of 10 is reserved for the IPFIX protocol specified in this document. Set ID values of 0 and 1 are not used for historical reasons [RFC3954]. The Set ID value of 2 is reserved for the Template Set. The Set ID value of 3 is reserved for the Options Template Set. All other Set ID values from 4 to 255 are reserved for future use. Set ID values above 255 are used for Data Sets.

New assignments in either IPFIX Version Number or IPFIX Set ID assignments require a Standards Action [RFC5226], i.e., they are to be made via Standards Track RFCs approved by the IESG.

Appendix A. IPFIX Encoding Examples

This appendix, which is a not a normative reference, contains IPFIX encoding examples.

Let’s consider the example of an IPFIX Message composed of a Template Set, a Data Set (which contains three Data Records), an Options Template Set and a Data Set (which contains 2 Data Records related to the previous Options Template Record).

IPFIX Message:

```
+--------+------------------------------------------. . .
|        | +--------------+ +------------------+
|Message | | Template     | | Data             |
| Header | | Set          | | Set              |   . . .
|        | | (1 Template) | | (3 Data Records) |
|        | +--------------+ +------------------+
+--------+------------------------------------------. . .

     . . .-------------------------------------------+
          +------------------+ +------------------+ |
          | Options          | | Data             | |
    . . . | Template Set     | | Set              | |
          | (1 Template)     | | (2 Data Records) | |
          +------------------+ +------------------+ |
     . . .-------------------------------------------+
```

A.1. Message Header Example

The Message Header is composed of:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Version = 0x000a          |         Length = 152          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Export Time                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sequence Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Observation Domain ID                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

A.2. Template Set Examples

A.2.1. Template Set Using IETF-Specified Information Elements

We want to report the following Information Elements:

- The IPv4 source IP address: sourceIPv4Address in [RFC5102bis], with a length of 4 octets

- The IPv4 destination IP address: destinationIPv4Address in [RFC5102bis], with a length of 4 octets

- The next-hop IP address (IPv4): ipNextHopIPv4Address in [RFC5102bis], with a length of 4 octets

- The number of packets of the Flow: packetDeltaCount in [RFC5102bis], with a length of 4 octets

- The number of octets of the Flow: octetDeltaCount in [RFC5102bis], with a length of 4 octets

Therefore, the Template Set will be composed of the following:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Set ID = 2            |      Length = 28 octets       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Template ID 256         |       Field Count = 5         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|    sourceIPv4Address = 8    |       Field Length = 4        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0| destinationIPv4Address = 12 |       Field Length = 4        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|  ipNextHopIPv4Address = 15  |       Field Length = 4        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|    packetDeltaCount = 2     |       Field Length = 4        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|    octetDeltaCount = 1      |       Field Length = 4        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

A.2.2. Template Set Using Enterprise-Specific Information Elements

We want to report the following Information Elements:

- The IPv4 source IP address: sourceIPv4Address in [RFC5102bis], with a length of 4 octets

- The IPv4 destination IP address: destinationIPv4Address in [RFC5102bis], with a length of 4 octets

- An enterprise-specific Information Element representing proprietary information, with a type of 15 and a length of 4

- The number of packets of the Flow: packetDeltaCount in [RFC5102bis], with a length of 4 octets

- The number of octets of the Flow: octetDeltaCount in [RFC5102bis], with a length of 4 octets

Therefore, the Template Set will be composed of the following:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Set ID = 2            |      Length = 32 octets       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Template ID 257         |       Field Count = 5         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|    sourceIPv4Address = 8    |       Field Length = 4        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0| destinationIPv4Address = 12 |       Field Length = 4        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1| Information Element Id. = 15|       Field Length = 4        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Enterprise number                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|    packetDeltaCount = 2     |       Field Length = 4        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|    octetDeltaCount = 1      |       Field Length = 4        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

A.3. Data Set Example

In this example, we report the following three Flow Records:

Src IP addr.	Dst IP addr.	Next Hop addr.	Packet Number	Octets Number
192.0.2.12	192.0.2.254	192.0.2.1	5009	5344385
192.0.2.27	192.0.2.23	192.0.2.2	748	388934
192.0.2.56	192.0.2.65	192.0.2.3	5	6534

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Set ID = 256         |          Length = 64          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          192.0.2.12                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          192.0.2.254                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          192.0.2.1                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             5009                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            5344385                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          192.0.2.27                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          192.0.2.23                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          192.0.2.2                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              748                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             388934                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          192.0.2.56                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          192.0.2.65                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          192.0.2.3                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               5                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              6534                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Note that padding is not necessary in this example.

A.4. Options Template Set Examples

A.4.1. Options Template Set Using IETF-Specified Information Elements

Per line card (the router being composed of two line cards), we want to report the following Information Elements:

- Total number of IPFIX Messages: exportedMessageTotalCount [RFC5102bis], with a length of 2 octets

- Total number of exported Flows: exportedFlowRecordTotalCount [RFC5102bis], with a length of 2 octets

The line card, which is represented by the lineCardId Information Element [RFC5102bis], is used as the Scope Field.

Therefore, the Options Template Set will be:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Set ID = 3            |          Length = 24          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Template ID 258         |        Field Count = 3        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Scope Field Count = 1     |0|     lineCardId = 141        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Scope 1 Field Length = 4    |0|exportedMessageTotalCount=41 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Field Length = 2        |0|exportedFlowRecordTotalCo.=42|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Field Length = 2        |           Padding             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

A.4.2. Options Template Set Using Enterprise-Specific Information Elements

Per line card (the router being composed of two line cards), we want to report the following Information Elements:

- Total number of IPFIX Messages: exportedMessageTotalCount [RFC5102bis], with a length of 2 octets

- An enterprise-specific number of exported Flows, with a type of 42 and a length of 4 octets

The line card, which is represented by the lineCardId Information

Element [RFC5102bis], is used as the Scope Field.

The format of the Options Template Set is as follows:

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |         Set ID = 3            |          Length = 28          |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |       Template ID 259         |        Field Count = 3        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |     Scope Field Count = 1     |0|     lineCardId = 141        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |   Scope 1 Field Length = 4    |0|exportedFlowRecordTotalCo.=41|
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |       Field Length = 2        |1|Information Element Id. = 42 |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |       Field Length = 4        |       Enterprise number      ...
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
...       Enterprise number      |           Padding             |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

A.4.3. Options Template Set Using an Enterprise-Specific Scope

In this example, we want to export the same information as in the example in Section A.4.1:

- Total number of IPFIX Messages: exportedMessageTotalCount [RFC5102bis], with a length of 2 octets

- Total number of exported Flows: exportedFlowRecordTotalCount [RFC5102bis], with a length of 2 octets

But this time, the information pertains to a proprietary scope, identified by enterprise-specific Information Element number 123.

The format of the Options Template Set is now as follows:

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |         Set ID = 3            |          Length = 28          |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |       Template ID 260         |        Field Count = 3        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |     Scope Field Count = 1     |1|Scope 1 Infor. El. Id. = 123 |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |    Scope 1 Field Length = 4   |       Enterprise Number      ...
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
...       Enterprise Number      |0|exportedMessageTotalCount=41 |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |       Field Length = 2        |0|exportedFlowRecordTotalCo.=42|
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |       Field Length = 2        |           Padding             |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

A.4.4. Data Set Using an Enterprise-Specific Scope

In this example, we report the following two Data Records:

Enterprise field 123	IPFIX Message	Exported Flow Records
1	345	10201
2	690	20402

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |      Set ID = 260             |         Length = 20           |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                               1                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |             345               |            10201              |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                               2                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |             690               |            20402              |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

A.5. Variable-Length Information Element Examples

A.5.1. Example of Variable-Length Information Element with Length Inferior to 255 Octets

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       5       |          5 octet Information Element          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

A.5.2. Example of Variable-Length Information Element with 3 Octet Length Encoding

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      255      |               1000            |    IE ...     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                1000 octet Information Element                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                              ...                              :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            ... IE             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

References

Normative References

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

[RFC3436]   Jungmaier, A., Rescorla, E., and M. Tuexen, "Transport Layer Security over Stream Control Transmission Protocol", RFC 3436, December 2002.

[RFC3492]   Costello, A., "Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA)", RFC 3492, March 2003.

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

[RFC4347]   Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security", RFC 4347, April 2006.

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

[RFC5226]   Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

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

[RFC5280]   Cooper, D., Santesson, S., Farrell, S. Boeyen, S. Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, April 2008.

[RFC5905]   Mills, D., Delaware, U., Martin, J., Burbank, J. and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010

[RFC5891]   J. Klensin, "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, August 2010.

[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-00.txt, Work in Progress, July 2011.

Informative References

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

[RFC1948] Bellovin, S., "Defending Against Sequence Number Attacks", RFC 1948, May 1996.

[RFC2579] McCloghrie, K., Perkins, D., and J. Schoenwaelder, "Textual Conventions for SMIv2", STD 58, RFC 2579, April 1999.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

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

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

[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

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

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

[RFC5471] Schmoll, C., Aitken, P., and B. Claise, "Guidelines for IP Flow Information Export (IPFIX) Testing", RFC5471, March 2009

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

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

[RFC6083] Tuexen, M., Seggelman, R. and E. Rescola, "Datagram Transport Layer Security (DTLS) for Stream Control

Transmission Protocol (SCTP)", RFC6083, January 2011.

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

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

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

[IPFIX-CONF] Muenz, G., Claise, B., and P. Aitken, "Configuration Data Model for IPFIX and PSAMP", draft-ietf-ipfix-configuration-model-10, Work in Progress, July 2011.

[IPFIX-MED-PROTO] Claise, B., Kobayashi, A., and B. Trammell, "Specification of the Protocol for IPFIX Mediations", draft-claise-ipfix-mediation-protocol-04, Work in Progress, July 2011.

[RFC5815bis] Dietz, T., Kobayashi, A., Claise, B., and G. Muenz, "Definitions of Managed Objects for IP Flow Information Export", draft-dkcm-ipfix-rfc5815bis-00.txt, Work in Progress, October 2011.

Acknowledgments

We would like to thank the following persons: Ganesh Sadasivan for his significant contribution during the initial phases of the protocol specification; Juergen Quittek for the coordination job within IPFIX and PSAMP; Nevil Brownlee, Dave Plonka, Paul Aitken, and Andrew Johnson for the thorough reviews; Randall Stewart and Peter Lei for their SCTP expertise and contributions; Martin Djernaes for the first essay on the SCTP section; Michael Behringer and Eric Vyncke for their advice and knowledge in security; Michael Tuexen for his help regarding the DTLS section; Elisa Boschi for her contribution regarding the improvement of SCTP sections; Mark Fullmer, Sebastian Zander, Jeff Meyer, Maurizio Molina, Carter Bullard, Tal Givoly, Lutz Mark, David Moore, Robert Lowe, Paul Calato, and many more, for the technical reviews and feedback.

Authors' Addresses

Benoit Claise
Cisco Systems
De Kleetlaan 6a b1
1831 Diegem
Belgium
Phone: +32 2 704 5622
EMail: bclaise@cisco.com

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
CH-8021 Zurich
Switzerland
Phone: +41 44 268 1536
EMail: simon.leinen@switch.ch

Thomas Dietz
NEC Europe Ltd.
NEC Laboratories Europe
Network Research Division
Kurfuersten-Anlage 36
69115 Heidelberg
Germany
Phone: +49 6221 4342-128
EMail: Thomas.Dietz@nw.neclab.eu

Brian Trammell
Swiss Federal Institute of Technology Zurich
Gloriastrasse 35
Zurich
Switzerland
Phone: +41 44 632 70 13
EMail: trammell@tik.ee.ethz.ch

Network Working Group T. Dietz, Ed.
Internet-Draft NEC Europe Ltd.
Intended status: Standards Track B. Claise
Expires: May 3, 2012 Cisco Systems, Inc.
J. Quittek
NEC Europe Ltd.
October 31, 2011

Definitions of Managed Objects for Packet Sampling
<draft-ietf-ipfix-psamp-mib-04.txt>

Abstract

This memo defines a portion of the Management Information Base (MIB) for use with network management protocols in the Internet community. In particular, it describes extensions to the IPFIX SELECTOR MIB module [I-D.dkcm-ipfix-rfc5815bis]. For IPFIX implementations that use packet Sampling (PSAMP) techniques as described in [RFC5475], this memo defines the PSAMP MIB module containing managed objects for providing information on applied packet selection functions and their parameters.

Status of this Memo

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 3, 2012.

Copyright Notice

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

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

Table of Contents

1. The Internet-Standard Management Framework

For a detailed overview of the documents that describe the current Internet-Standard Management Framework, please refer to section 7 of RFC 3410 [RFC3410].

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. MIB objects are generally accessed through the Simple Network Management Protocol (SNMP). Objects in the MIB are defined using the mechanisms defined in the Structure of Management Information (SMI). This memo specifies MIB modules that is compliant to the SMIv2, which is described in STD 58, RFC 2578 [RFC2578], STD 58, RFC 2579 [RFC2579] and STD 58,RFC 2580 [RFC2580].

2. Introduction

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

This document is a product of the IP Flow Information eXport (IPFIX) working group. Work on this document was started in the Packet Sampling (PSAMP) Working Group (WG) and moved to the IPFIX WG when the PSAMP WG was concluded.

Its purpose is to define managed objects for monitoring PSAMP Devices performing packet selection by Sampling and Filtering as described in [RFC5475].

It is assumed that packet Sampling is performed according to the framework defined in [RFC5474].

Managed objects in the PSAMP MIB module are defined as an extension of the IPFIX MIB and IPFIX SELECTOR MIB modules [I-D.dkcm-ipfix-rfc5815bis]. Since the IPFIX MIB module is only for monitoring the same holds true for the PSAMP MIB module defined in this document. The definition of objects is in line with the PSAMP information model [RFC5477].

Section 3 gives an overview of the PSAMP documents, while section 4 refers to the related IPFIX documents. Section 5 describes the structure of the PSAMP MIB module and section 6 contains the formal definition. Security issues are discussed in section 7.

3. PSAMP Documents Overview

[RFC5474]: "A Framework for Packet Selection and Reporting" describes the PSAMP framework for network elements to select subsets of packets by statistical and other methods, and to export a stream of reports on the selected packets to a Collector.

[RFC5475]: "Sampling and Filtering Techniques for IP Packet Selection" describes the set of packet selection techniques supported by PSAMP.

[RFC5476]: "Packet Sampling (PSAMP) Protocol Specifications" specifies the export of packet information from a PSAMP Exporting Process to a PSAMP Collecting Process.

[RFC5477]: "Information Model for Packet Sampling Exports" defines an information and data model for PSAMP.

This document: "Definitions of Managed Objects for Packet Sampling" describes the PSAMP Management Information Base.

4. Related IPFIX Documents

The IPFIX protocol provides network administrators with access to IP Flow information.

[RFC5101]: The protocol document "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information" specifies how IPFIX Data Records and Templates are carried via a congestion-aware transport protocol from IPFIX Exporting Processes to IPFIX Collecting Processes. It also specifies the data types used in the PSAMP MIB module and their encoding.

[I-D.dkcm-ipfix-rfc5815bis]: The IPFIX MIB "Definitions of Managed Objects for IP Flow Information Export" is the basis for this document because it extends the IPFIX SELECTOR MIB defined there.

5. Structure of the PSAMP MIB module

The IPFIX MIB module defined in [I-D.dkcm-ipfix-rfc5815bis] has the concept of a packet Selection Process containing a set of Selector function instances. Selection Processes and functions are referenced in the ipfixSelectionProcessTable of the IPFIX MIB module. The ipfixSelectionProcessTable identifies an instance of a Selector function by an OID. The OID points to an object that describes the Selector function. For simple Selector functions without parameters, the OID refers to an object which contains only one additional object indicating the current availability of the function. For functions which have one or more parameters the object has a subtree that in addition to an availability object contains a table with a conceptual column for each parameter. Entries (conceptual rows) in this table represent different combinations of parameter values for instances of the Selector function.

The object ipfixSelectorFunctions in the IPFIX SELECTOR MIB module serves as the root for objects that describe instances of packet Selector functions. The IPFIX SELECTOR MIB is a very small module which is defined in [I-D.dkcm-ipfix-rfc5815bis]. The top level OIDs of the parameter trees located beneath ipfixSelectorFunctions are maintained by IANA. In the IPFIX SELECTOR MIB module as defined by [I-D.dkcm-ipfix-rfc5815bis] the object ipfixSelectorFunctions contains just a single trivial packet Selector function called ipfixFuncSelectAll that selects every packet and has no parameter:

```
ipfixSelectorMIB
+- ipfixSelectorObjects(1)
   +- ipfixSelectorFunctions(1)
      +- ipfixFuncSelectAll(1)
         +- ipfixFuncSelectAllAvail(1)
```

The PSAMP MIB module defined in this document registers additional toplevel OIDs for the parameter subtrees of its Selector functions in the IPFIX-SELECTOR-MIB Function subregistry according to the procedures defined in [I-D.dkcm-ipfix-rfc5815bis]. It introduces six new subtrees beneath ipfixSelectorFunctions. Each of them describes a packet Selector function with one or more parameters. Naming and ordering of objects is fully in line with the guidelines given in section 6.1 of [I-D.dkcm-ipfix-rfc5815bis]. All functions and their parameters are already listed in the overview of functions given by the table in section 8.2.1 of [RFC5477].

5.1. Textual Conventions

The PSAMP MIB module imports two textual conventions which define data types used in this MIB from other MIB modules. The Unsigned64TC data type is imported from the APPLICATION MIB [RFC2564] and the Float64TC data type is imported from the FLOAT-TC-MIB [RFC6340]. Those data types are defined according to [RFC5101]. Those data types are not an integral part of [RFC2578] but are needed to define objects in this MIB module that conform to the Information Elements defined for those objects in [RFC5477].

The Unsigned64TC textual convention describes an unsigned integer of 64 bits. It is imported from the APPLICATION MIB. The Float64TC textual convention describes the format that is used for 64 bit floating point numbers.

5.2. Packet Selection Functions

In general, different packet Selector functions have different parameters. The PSAMP MIB module contains six objects with subtrees that provide information on parameters of function instances of different Selector functions. All objects are named and structured according to section 8.2.1 of [RFC5477]:

```
ipfixSelectorFunctions(1)
+-- psampSampCountBased(2)
+- -psampSampTimeBased(3)
+-- psampSampRandOutOfN(4)
+-- psampSampUniProb(5)
+-- psampFiltPropMatch(6)
+-- psampFiltHash(7)
```

Indexing of these functions in the PSAMP MIB module starts with index (2). The function ipfixFuncSelectAll with index (1) is already defined in the IPFIX SELECTOR MIB module as shown above.

The object tree for each of these functions is described below. Semantics of all functions and their parameters are described in detail in [RFC5475]. More information on the Selector Reports can also be found in section 6.5.2 of [RFC5476].

5.2.1. Systematic Count-based Sampling

The first Selector function is systematic count-based Sampling. Its availability is indicated by object psampSampCountBasedAvail. The function has two parameters: psampSampCountBasedInterval and psampSampCountBasedSpace. Different combination of values of these parameters for different instances of the Selector function are represented by different conceptual rows in table psampSampCountBasedParamSetEntry:

```
psampSampCountBased(2)
+-- psampSampCountBasedAvail(1)
+-- psampSampCountBasedParamSetTable(2)
   +-- psampSampCountBasedParamSetEntry(1) [psampSampCountBasedIndex]
      +-- psampSampCountBasedIndex(1)
      +-- psampSampCountBasedInterval(2)
      +-- psampSampCountBasedSpace(3)
```

5.2.2. Systematic Time-based Sampling

The second Selector function is systematic time-based Sampling. The structure of the sub-tree for this function is similar to the psampSampCountBased sub-tree. Parameters are psampSampTimeBasedInterval and psampSampTimeBasedSpace. They appear to be the same as for count based Sampling, but their data types are different because they indicate time values instead of numbers of packets:

```
psampSampTimeBased(3)
+-- psampSampTimeBasedAvail(1)
+-- psampSampTimeBasedParamSetTable(2)
   +-- psampSampTimeBasedParamSetEntry(1) [psampSampTimeBasedIndex]
      +-- psampSampTimeBasedIndex(1)
      +-- psampSampTimeBasedInterval(2)
      +-- psampSampTimeBasedSpace(3)
```

5.2.3. Random n-out-of-N Sampling

The third Selector function is random n-out-of-N Sampling. Parameters are psampSampRandOutOfNSize and psampSampRandOutOfNPopulation:

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

5.2.4. Uniform Probabilistic Sampling

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

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

5.2.5. Property Match Filtering

The fifth Selector function is property match Filtering. For this Selector function there is a broad variety of possible parameters that could be used. But as stated in section 8.2.1 of [RFC5477] there are no agreed parameters specified and the sub-tree for this function only contains an object indicating the availability of this function. Parameters cannot be retrieved via the PSAMP MIB module:

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

5.2.6. Hash-based Filtering

The sixth Selector function is hash-based Filtering. The object psampFiltHashFunction is an enumeration that specifies the kind of hash function that is applied. These hash function have quite a number of parameters and the actual number may vary with the choice of the hash function applied. The common parameter set for all hash-based Filtering functions contains 7 parameters: psampFiltHashInitializerValue, psampFiltHashIpPayloadOffset, psampFiltHashIpPayloadSize, psampFiltHashSelectedRangeMin, psampFiltHashSelectedRangeMax, psampFiltHashOutputRangeMin, and psampFiltHashOutputRangeMax.

```
psampFiltHash(7)
+-- psampFiltHashAvail(1)
+-- psampFiltHashCapabilities(2)
+-- psampFiltHashParamSetTable(3)
   +-- psampFiltHashParamSetEntry(1) [psampFiltHashIndex]
      +-- psampFiltHashIndex(1)
      +-- psampFiltHashFunction(2)
      +-- psampFiltHashInitializerValue(3)
      +-- psampFiltHashIpPayloadOffset(4)
      +-- psampFiltHashIpPayloadSize(5)
      +-- psampFiltHashSelectedRangeMin(6)
      +-- psampFiltHashSelectedRangeMax(7)
      +-- psampFiltHashOutputRangeMin(8)
      +-- psampFiltHashOutputRangeMax(9)
```

Further parameters depend on the applied hash function and are not specified within the PSAMP MIB module.

6. Definitions

```
PSAMP-MIB DEFINITIONS ::= BEGIN
```

```
   IMPORTS
       MODULE-IDENTITY, OBJECT-TYPE, Integer32, Unsigned32, mib-2
           FROM SNMPv2-SMI                  -- RFC2578
       TruthValue
           FROM SNMPv2-TC                   -- RFC2579
       MODULE-COMPLIANCE, OBJECT-GROUP
           FROM SNMPv2-CONF                 -- RFC2580
       Unsigned64TC
           FROM APPLICATION-MIB             -- RFC2564
       Float64TC
           FROM FLOAT-TC-MIB                -- RFC6340
       ipfixSelectorFunctions
           FROM IPFIX-SELECTOR-MIB;

   psampMIB MODULE-IDENTITY
       LAST-UPDATED "201110311200Z"         -- 31 October 2011
       ORGANIZATION "IETF IPFIX Working Group"
       CONTACT-INFO
           "WG charter:
             http://www.ietf.org/html.charters/ipfix-charter.html

           Mailing Lists:
             General Discussion: ipfix@ietf.org
             To Subscribe: http://www1.ietf.org/mailman/listinfo/ipfix
             Archive:
         http://www1.ietf.org/mail-archive/web/ipfix/current/index.html

           Editor:
             Thomas Dietz
             NEC Europe Ltd.
             NEC Laboratories Europe
             Network Research Division
             Kurfuersten-Anlage 36
             69115 Heidelberg
             Germany
             Phone: +49 6221 4342-128
             Email: Thomas.Dietz@neclab.eu

             Benoit Claise
             Cisco Systems, Inc.
             De Kleetlaan 6a b1
             Degem 1831
             Belgium
             Phone:  +32 2 704 5622
             Email: bclaise@cisco.com

             Juergen Quittek
             NEC Europe Ltd.
```

```
             NEC Laboratories Europe
             Network Research Division
             Kurfuersten-Anlage 36
             69115 Heidelberg
             Germany
             Phone: +49 6221 4342-115
             Email: quittek@neclab.eu"
           DESCRIPTION
           "The PSAMP MIB defines managed objects for packet sampling
           and filtering.
           These objects provide information about managed nodes
           supporting packet sampling, including packet sampling
           capabilities, configuration and statistics.
           The PSAMP MIB module registers additional toplevel OIDs for
           the parameter subtrees of its Selector functions in the
           IPFIX-SELECTOR-MIB Function subregistry according to the
           procedures defined in RFC 5815bis.
   -- RFC Ed.: replace RFC 5815bis above with the actual RFC number
   --          once assigned and remove this notice

           Copyright (c) 2011 IETF Trust and the persons identified as
           the document authors.  All rights reserved. This version
           of this MIB module is part of RFC yyyy; see the RFC itself
           for full legal notices"
   -- RFC Ed.: replace yyyy with actual RFC number & remove this notice
        --  Revision history

        REVISION     "201110311200Z"         -- 31 October 2011
        DESCRIPTION
            "Initial version, published as RFC yyyy."
   -- RFC Ed.: replace yyyy with actual RFC number & remove this notice

       ::= { mib-2 xxx }
   -- RFC Ed.: replace xxx which is to be assigned by IANA & remove
   -- this notice.

   -- Top level structure of the MIB

   psampObjects     OBJECT IDENTIFIER ::= { psampMIB 1 }
   psampConformance OBJECT IDENTIFIER ::= { psampMIB 2 }


   --==================================================================
   -- Packet selection sampling methods group of objects
   --==================================================================

   --==================================================================
   --* Method 1: Systematic count-based Sampling
```

```
--==================================================================

-- Reference: RFC5475, Section 5.1, RFC5476 Section 6.5.2.1 and
--            RFC5477, Section 8.2
psampSampCountBased OBJECT IDENTIFIER
    ::= { ipfixSelectorFunctions 2 }

psampSampCountBasedAvail OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This object indicates the availability of systematic
        count-based sampling at the managed node.

        A Selector may be unavailable if it is implemented but
        currently disabled due to e.g., administrative reasons, lack
        of resources or similar."
    DEFVAL { false }
    ::= { psampSampCountBased 1 }

-- Parameter Set Table +++++++++++++++++++++++++++++++++++++++++++++

psampSampCountBasedParamSetTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF
                PsampSampCountBasedParamSetEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This table lists configurations of systematic count-based
        packet sampling.  A parameter set describing a
        configuration contains two parameters: the sampling
        interval length and space."
    ::= { psampSampCountBased 2 }

psampSampCountBasedParamSetEntry OBJECT-TYPE
    SYNTAX      PsampSampCountBasedParamSetEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Defines an entry in the psampSampCountBasedParamSetTable."
    INDEX { psampSampCountBasedIndex }
    ::= { psampSampCountBasedParamSetTable 1 }

PsampSampCountBasedParamSetEntry ::=
    SEQUENCE {
        psampSampCountBasedIndex     Integer32,
        psampSampCountBasedInterval  Unsigned32,
```

```
          psampSampCountBasedSpace     Unsigned32
      }

   psampSampCountBasedIndex OBJECT-TYPE
       SYNTAX      Integer32 (1..2147483647)
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "The index of this parameter set in the
           psampSampCountBasedParamSetTable. It is used in the
           object ipfixSelectionProcessSelectorFunction entries of
           the ipfixSelectionProcessTable in the IPFIX-MIB as reference
           to this parameter set."
       ::= { psampSampCountBasedParamSetEntry 1 }

   psampSampCountBasedInterval OBJECT-TYPE
       SYNTAX      Unsigned32
       UNITS       "packets"
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "This object specifies the number of packets that are
           consecutively sampled.  A value of 100 means that 100
           consecutive packets are sampled."
       REFERENCE
           "RFC5475, Section 5.1 and RFC5477, Section 8.2"
       ::= { psampSampCountBasedParamSetEntry 2 }

   psampSampCountBasedSpace OBJECT-TYPE
       SYNTAX      Unsigned32
       UNITS       "packets"
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "This object specifies the number of packets between two
           psampSampCountBasedInterval's.  A value of 100 means that
           the next interval starts 100 packets (which are not sampled)
           after the current psampSampCountBasedInterval is over."
       REFERENCE
           "RFC5475, Section 5.1 and RFC5477, Section 8.2"
       ::= { psampSampCountBasedParamSetEntry 3 }

   --===================================================================
   --* Method 2: Systematic time-based Sampling
   --===================================================================

   -- Reference: RFC5475, Section 5.1, RFC5476 Section 6.5.2.2 and
   --            RFC5477, Section 8.2
```

```
   psampSampTimeBased OBJECT IDENTIFIER
       ::= { ipfixSelectorFunctions 3 }

   psampSampTimeBasedAvail OBJECT-TYPE
       SYNTAX      TruthValue
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "This object indicates the availability of systematic
           time-based sampling at the managed node.

           A Selector may be unavailable if it is implemented but
           currently disabled due to e.g., administrative reasons, lack
           of resources or similar."
       DEFVAL { false }
       ::= { psampSampTimeBased 1 }

   -- Parameter Set Table ++++++++++++++++++++++++++++++++++++++++++++

   psampSampTimeBasedParamSetTable OBJECT-TYPE
       SYNTAX      SEQUENCE OF
                   PsampSampTimeBasedParamSetEntry
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "This table lists configurations of systematic time-based
           packet sampling. A parameter set describing a configuration
           contains two parameters: the sampling interval length and
           the space."
       ::= { psampSampTimeBased 2 }

   psampSampTimeBasedParamSetEntry OBJECT-TYPE
       SYNTAX      PsampSampTimeBasedParamSetEntry
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "Defines an entry in the psampSampTimeBasedParamSetTable."
       INDEX { psampSampTimeBasedIndex }
       ::= { psampSampTimeBasedParamSetTable 1 }

   PsampSampTimeBasedParamSetEntry ::=
       SEQUENCE {
           psampSampTimeBasedIndex     Integer32,
           psampSampTimeBasedInterval  Unsigned32,
           psampSampTimeBasedSpace     Unsigned32
       }

   psampSampTimeBasedIndex OBJECT-TYPE
```

```
        SYNTAX      Integer32 (1..2147483647)
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "The index of this parameter set in the
            psampSampTimeBasedParamSetTable. It is used in the
            object ipfixSelectionProcessSelectorFunction entries of
            the ipfixSelectionProcessTable in the IPFIX-MIB as reference
            to this parameter set."
        ::= { psampSampTimeBasedParamSetEntry 1 }

    psampSampTimeBasedInterval OBJECT-TYPE
        SYNTAX      Unsigned32
        UNITS       "microseconds"
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "This object specifies the time interval in microseconds
            during which all arriving packets are sampled."
        REFERENCE
            "RFC5475, Section 5.1 and RFC5477, Section 8.2"
        ::= { psampSampTimeBasedParamSetEntry 2 }

    psampSampTimeBasedSpace OBJECT-TYPE
        SYNTAX      Unsigned32
        UNITS       "microseconds"
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "This object specifies the time interval in microseconds
            between two psampSampTimeBasedInterval's.  A value of 100
            means that the next interval starts 100 microseconds (during
            which no packets are sampled) after the current
            psampSampTimeBasedInterval is over."
        REFERENCE
            "RFC5475, Section 5.1 and RFC5477, Section 8.2"
        ::= { psampSampTimeBasedParamSetEntry 3 }

    --==================================================================
    --* Method 3: Random n-out-of-N Sampling
    --==================================================================

    -- Reference: RFC5475, Section 5.2.1, RFC5476 Section 6.5.2.3 and
    --            RFC5477, Section 8.2
    psampSampRandOutOfN OBJECT IDENTIFIER
        ::= { ipfixSelectorFunctions 4 }

    psampSampRandOutOfNAvail OBJECT-TYPE
```

```
        SYNTAX      TruthValue
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "This object indicates the availability of random n-out-of-N
            sampling at the managed node.

            A Selector may be unavailable if it is implemented but
            currently disabled due to e.g., administrative reasons, lack
            of resources or similar."
        DEFVAL { false }
        ::= { psampSampRandOutOfN 1 }

    -- Parameter Set Table +++++++++++++++++++++++++++++++++++++++++++++

    psampSampRandOutOfNParamSetTable OBJECT-TYPE
        SYNTAX      SEQUENCE OF
                    PsampSampRandOutOfNParamSetEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "This table lists configurations of random n-out-of-N
            sampling.  A parameter set describing a configuration
            contains two parameters, the sampling size and the
            parent population."
        ::= { psampSampRandOutOfN 2 }

    psampSampRandOutOfNParamSetEntry OBJECT-TYPE
        SYNTAX      PsampSampRandOutOfNParamSetEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "Defines an entry in the psampSampRandOutOfNParamSetTable."
        INDEX { psampSampRandOutOfNIndex }
        ::= { psampSampRandOutOfNParamSetTable 1 }

    PsampSampRandOutOfNParamSetEntry ::=
        SEQUENCE {
            psampSampRandOutOfNIndex        Integer32,
            psampSampRandOutOfNSize Unsigned32,
            psampSampRandOutOfNPopulation   Unsigned32
        }

    psampSampRandOutOfNIndex OBJECT-TYPE
        SYNTAX      Integer32 (1..2147483647)
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
```

```
           "The index of this parameter set in the
           psampSampRandOutOfNParamSetTable.  It is used in the
           object ipfixSelectionProcessSelectorFunction entries of
           the ipfixSelectionProcessTable in the IPFIX-MIB as reference
           to this parameter set."
       ::= { psampSampRandOutOfNParamSetEntry 1 }

   psampSampRandOutOfNSize OBJECT-TYPE
       SYNTAX      Unsigned32
       UNITS       "packets"
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "This object specifies the number of elements taken from the
           parent Population specified in
           psampSampRandOutOfNPopulation."
       REFERENCE
           "RFC5475, Section 5.2.1 and RFC5477, Section 8.2"
       ::= { psampSampRandOutOfNParamSetEntry 2 }

   psampSampRandOutOfNPopulation OBJECT-TYPE
       SYNTAX      Unsigned32
       UNITS       "packets"
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "This object specifies the number of elements in the parent
           Population."
       REFERENCE
           "RFC5475, Section 5.2.1 and RFC5477, Section 8.2"
       ::= { psampSampRandOutOfNParamSetEntry 3 }

   --==================================================================
   --* Method 4: Uniform probabilistic Sampling
   --==================================================================

   -- Reference: RFC5475, Section 5.2.2, RFC5476 Section 6.5.2.4 and
   --            RFC5477, Section 8.2
   psampSampUniProb OBJECT IDENTIFIER ::= { ipfixSelectorFunctions 5 }

   psampSampUniProbAvail OBJECT-TYPE
       SYNTAX      TruthValue
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "This object indicates the availability of random uniform
           probabilistic sampling at the managed node.
```

```
          A Selector may be unavailable if it is implemented but
          currently disabled due to e.g., administrative reasons, lack
          of resources or similar."
      DEFVAL { false }
      ::= { psampSampUniProb 1 }

   -- Parameter Set Table ++++++++++++++++++++++++++++++++++++++++++++

   -- Reference: RFC5475, Section 5.2.2.1 and RFC5477, Section 8.2
   psampSampUniProbParamSetTable OBJECT-TYPE
       SYNTAX      SEQUENCE OF
                   PsampSampUniProbParamSetEntry
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "This table lists configurations of random probabilistic
           sampling.  A parameter set describing a configuration
           contains a single parameter only: the sampling probability."
       ::= { psampSampUniProb 2 }

   psampSampUniProbParamSetEntry OBJECT-TYPE
       SYNTAX      PsampSampUniProbParamSetEntry
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "Defines an entry in the psampSampUniProbParamSetTable."
       INDEX { psampSampUniProbIndex }
       ::= { psampSampUniProbParamSetTable 1 }

   PsampSampUniProbParamSetEntry ::=
       SEQUENCE {
           psampSampUniProbIndex       Integer32,
           psampSampUniProbProbability Float64TC
       }

   psampSampUniProbIndex OBJECT-TYPE
       SYNTAX      Integer32 (1..2147483647)
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "The index of this parameter set in the
           psampSampUniProbParamSetTable.  It is used in the
           object ipfixSelectionProcessSelectorFunction entries of
           the ipfixSelectionProcessTable in the IPFIX-MIB as reference
           to this parameter set."
       ::= { psampSampUniProbParamSetEntry 1 }

   psampSampUniProbProbability OBJECT-TYPE
```

```
       SYNTAX      Float64TC
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "This object specifies the probability that a packet is
           sampled, expressed as a value between 0 and 1.  The
           probability is equal for every packet.  A value of 0 means
           no packet is sampled since the probability is 0. A value
           of 1 means all packets are sampled since the
           probability is 1. NaN (not a number) and infinity MUST NOT
           be used."
       REFERENCE
           "RFC5475, Section 5.2.2.1 and RFC5477, Section 8.2"
       ::= { psampSampUniProbParamSetEntry 2 }

   --==================================================================
   -- Packet selection filtering methods group of objects
   --==================================================================

   --==================================================================
   --* Method 5: Property Match filtering
   --==================================================================

   -- Reserves Method 5 (see RFC5475, Section 6.1, RFC5476
   -- Section 6.5.2.5 and RFC5477)
   psampFiltPropMatch OBJECT IDENTIFIER
       ::= { ipfixSelectorFunctions 6 }

   psampFiltPropMatchAvail OBJECT-TYPE
       SYNTAX      TruthValue
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "This object indicates the availability of property match
           filtering at the managed node.

           A Selector may be unavailable if it is implemented but
           currently disabled due to e.g., administrative reasons, lack
           of resources or similar."
       DEFVAL { false }
       ::= { psampFiltPropMatch 1 }

   --==================================================================
   --* Method 6: Hash filtering
   --==================================================================

   -- Reference: RFC5475, Section 6.2, RFC5476 Section 6.5.2.6 and
   --            RFC5477, Section 8.3
```

```
   psampFiltHash OBJECT IDENTIFIER ::= { ipfixSelectorFunctions 7 }

   psampFiltHashAvail OBJECT-TYPE
       SYNTAX      TruthValue
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "This object indicates the availability of hash filtering
           at the managed node.

           A Selector may be unavailable if it is implemented but
           currently disabled due to e.g., administrative reasons, lack
           of resources or similar."
       DEFVAL { false }
       ::= { psampFiltHash 1 }

   psampFiltHashCapabilities OBJECT IDENTIFIER
       ::= { psampFiltHash 2 }

   -- Parameter Set Table +++++++++++++++++++++++++++++++++++++++++++++

   -- Reference: RFC5475, Sections 6.2, 3.8, and 7.1
   psampFiltHashParamSetTable OBJECT-TYPE
       SYNTAX      SEQUENCE OF
                   PsampFiltHashParamSetEntry
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "This table lists configurations of hash filtering. A
           parameter set describing a configuration contains eight
           parameters describing the hash function."
       ::= { psampFiltHash 3 }

   psampFiltHashParamSetEntry OBJECT-TYPE
       SYNTAX      PsampFiltHashParamSetEntry
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "Defines an entry in the psampFiltHashParamSetTable."
       INDEX { psampFiltHashIndex }
       ::= { psampFiltHashParamSetTable 1 }

   PsampFiltHashParamSetEntry ::=
       SEQUENCE {
           psampFiltHashIndex            Integer32,
           psampFiltHashFunction         INTEGER,
           psampFiltHashInitializerValue Unsigned64TC,
           psampFiltHashIpPayloadOffset  Unsigned64TC,
```

```
            psampFiltHashIpPayloadSize    Unsigned64TC,
            psampFiltHashSelectedRangeMin Unsigned64TC,
            psampFiltHashSelectedRangeMax Unsigned64TC,
            psampFiltHashOutputRangeMin   Unsigned64TC,
            psampFiltHashOutputRangeMax   Unsigned64TC
        }

   psampFiltHashIndex OBJECT-TYPE
        SYNTAX      Integer32 (1..2147483647)
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "The index of this parameter set in the
            psampFiltHashParamSetTable. It is used in the
            object ipfixSelectionProcessSelectorFunction entries of
            the ipfixSelectionProcessTable in the IPFIX-MIB as reference
            to this parameter set."
        ::= { psampFiltHashParamSetEntry 1 }

   psampFiltHashFunction OBJECT-TYPE
        SYNTAX      INTEGER {
                        crc32(1),
                        ipsx(2),
                        bob(3)
                    }
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The Hash Function used by this filter. The PSAMP-MIB
            defines the following Hash Functions:

            crc32(1): The CRC32 Hash Function as defined in RFC1141.

            ipsx(2): The IPSX Hash Function as described in RFC5475
                appendix A.1.

            bob(3): The BOB Hash Function as described in RFC5475
                appendix A.2.
            "
        REFERENCE
            "RFC5475, Section 6.2 and Appendixes A.1 and A.2."
        ::= { psampFiltHashParamSetEntry 2 }

   psampFiltHashInitializerValue OBJECT-TYPE
        SYNTAX      Unsigned64TC
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
```

```
            "This object specifies the initializer value to the hash
            function."
        REFERENCE
            "RFC5475, Sections 6.2, 3.8, and 7.1"
        ::= { psampFiltHashParamSetEntry 3 }

    psampFiltHashIpPayloadOffset OBJECT-TYPE
        SYNTAX      Unsigned64TC
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "This object specifies the IP payload offset used by a
            Hash-based Selection Selector."
        REFERENCE
            "RFC5475, Sections 6.2, 3.8, and 7.1"
        ::= { psampFiltHashParamSetEntry 4 }

    psampFiltHashIpPayloadSize OBJECT-TYPE
        SYNTAX      Unsigned64TC
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "This object specifies the IP payload size used by a
            Hash-based Selection Selector."
        REFERENCE
            "RFC5475, Sections 6.2, 3.8, and 7.1"
        ::= { psampFiltHashParamSetEntry 5 }

    psampFiltHashSelectedRangeMin OBJECT-TYPE
        SYNTAX      Unsigned64TC
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "This object specifies the value for the beginning of a hash
            function's selected range."
        REFERENCE
            "RFC5475, Sections 6.2, 3.8, and 7.1"
        ::= { psampFiltHashParamSetEntry 6 }

    psampFiltHashSelectedRangeMax OBJECT-TYPE
        SYNTAX      Unsigned64TC
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "This object specifies the value for the end of a hash
            function's selected range."
        REFERENCE
            "RFC5475, Sections 6.2, 3.8, and 7.1"
```

```
       ::= { psampFiltHashParamSetEntry 7 }

   psampFiltHashOutputRangeMin OBJECT-TYPE
       SYNTAX      Unsigned64TC
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "This object specifies the value for the beginning of a hash
           function's potential output range."
       REFERENCE
           "RFC5475, Sections 6.2, 3.8, and 7.1"
       ::= { psampFiltHashParamSetEntry 8 }

   psampFiltHashOutputRangeMax OBJECT-TYPE
       SYNTAX      Unsigned64TC
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "This object specifies the value for the end of a hash
           function's potential output range."
       REFERENCE
           "RFC5475, Sections 6.2, 3.8, and 7.1"
       ::= { psampFiltHashParamSetEntry 9 }

   --==================================================================
   -- Conformance information
   --==================================================================

   psampCompliances OBJECT IDENTIFIER ::= { psampConformance 1 }
   psampGroups      OBJECT IDENTIFIER ::= { psampConformance 2 }

   --==================================================================
   -- Compliance statements
   --==================================================================

   psampCompliance MODULE-COMPLIANCE
       STATUS  current
       DESCRIPTION
           "The implementation of all objects is optional and depends
           on the implementation of the corresponding functionality in
           the equipment."
       MODULE  -- this module
           GROUP psampGroupSampCountBased
           DESCRIPTION
               "These objects must be implemented if systematic
               count-based sampling is implemented in the equipment."
           GROUP psampGroupSampTimeBased
           DESCRIPTION
```

```
               "These objects must be implemented if systematic
               time-based sampling is implemented in the equipment."
           GROUP psampGroupSampRandOutOfN
           DESCRIPTION
               "These objects must be implemented if random n-out-of-N
               sampling is implemented in the equipment."
           GROUP psampGroupSampUniProb
           DESCRIPTION
               "These objects must be implemented if uniform
               probabilistic sampling is implemented in the equipment."
           GROUP psampGroupFiltPropMatch
           DESCRIPTION
               "These objects must be implemented if the property match
               filtering is implemented in the equipment."
           GROUP psampGroupFiltHash
           DESCRIPTION
               "These objects must be implemented if hash filtering
               is implemented in the equipment."
       ::= { psampCompliances 1 }

   --==================================================================
   -- MIB groupings
   --==================================================================

   psampGroupSampCountBased OBJECT-GROUP
       OBJECTS {
                 psampSampCountBasedAvail,
                 psampSampCountBasedInterval,
                 psampSampCountBasedSpace
               }
       STATUS  current
       DESCRIPTION
           "These objects are needed if count based sampling is
           implemented."
       ::= { psampGroups 1 }

   psampGroupSampTimeBased OBJECT-GROUP
       OBJECTS {
                 psampSampTimeBasedAvail,
                 psampSampTimeBasedInterval,
                 psampSampTimeBasedSpace
               }
       STATUS  current
       DESCRIPTION
           "These objects are needed if time based sampling is
           implemented."
       ::= { psampGroups 2 }
```

```
   psampGroupSampRandOutOfN OBJECT-GROUP
       OBJECTS {
                 psampSampRandOutOfNAvail,
                 psampSampRandOutOfNSize,
                 psampSampRandOutOfNPopulation
               }
       STATUS  current
       DESCRIPTION
           "These objects are needed if random n-out-of-N sampling is
           implemented."
       ::= { psampGroups 3 }

   psampGroupSampUniProb OBJECT-GROUP
       OBJECTS {
                 psampSampUniProbAvail,
                 psampSampUniProbProbability
               }
       STATUS  current
       DESCRIPTION
           "These objects are needed if uniform probabilistic sampling
           is implemented."
       ::= { psampGroups 4 }

   psampGroupFiltPropMatch OBJECT-GROUP
       OBJECTS {
                 psampFiltPropMatchAvail
               }
       STATUS  current
       DESCRIPTION
           "These objects are needed if property match filtering is
           implemented."
       ::= { psampGroups 5 }

   psampGroupFiltHash OBJECT-GROUP
       OBJECTS {
                 psampFiltHashAvail,
                 psampFiltHashFunction,
                 psampFiltHashInitializerValue,
                 psampFiltHashIpPayloadOffset,
                 psampFiltHashIpPayloadSize,
                 psampFiltHashSelectedRangeMin,
                 psampFiltHashSelectedRangeMax,
                 psampFiltHashOutputRangeMin,
                 psampFiltHashOutputRangeMax
               }
       STATUS  current
       DESCRIPTION
           "These objects are needed if hash filtering is implemented."
```

```
       ::= { psampGroups 6 }

   END
```

7. Security Considerations

There are no management objects defined in this MIB module that have a MAX-ACCESS clause of read-write and/or read-create. So, if this MIB module is implemented correctly, then there is no risk that an intruder can alter or create any management objects of this MIB module via direct SNMP SET operations.

All tables in this MIB module may be considered sensitive or vulnerable in some network environments because objects in the tables may reveal information about the network infrastructure and device configuration. It is thus important to control even GET and/or NOTIFY access to these objects and possibly to even encrypt the values of these objects when sending them over the network via SNMP.

SNMP versions prior to SNMPv3 did not include adequate security. Even if the network itself is secure (for example by using IPsec), there is no control as to who on the secure network is allowed to access and GET/SET (read/change/create/delete) the objects in this MIB module.

It is RECOMMENDED that implementers consider the security features provided by the SNMPv3 framework (see [RFC3410], section 8), including full support for the SNMPv3 cryptographic mechanisms (for authentication and privacy).

Further, deployment of SNMP versions prior to SNMPv3 is NOT RECOMMENDED. Instead, it is RECOMMENDED to deploy SNMPv3 and to enable cryptographic security. It is then a customer/operator responsibility to ensure that the SNMP entity giving access to an instance of this MIB module is properly configured to give access to the objects only to those principals (users) who have legitimate rights to GET or SET (change/create/delete) them.

8. IANA Considerations

The MIB module in this document uses the following IANA-assigned OBJECT IDENTIFIER values recorded in the SMI Numbers registry:

Descriptor	OBJECT IDENTIFIER value
psampMIB	{ mib-2 xxx }

Further on, IANA will register the following toplevel OIDs in the IPFIX-SELECTOR-MIB Functions sub-registry at http://www.iana.org/assignments/smi-numbers according to the procedures set forth in [I-D.dkcm-ipfix-rfc5815bis]:

Decimal	Name	Description	Reference
2	psampSampCountBased	Count based sampling	[RFCyyyy]
3	psampSampTimeBased	Time based sampling	[RFCyyyy]
4	psampSampRandOutOfN	Random n-out-of-N sampling	[RFCyyyy]
5	psampSampUniProb	Universal probabilistic samp.	[RFCyyyy]
6	psampFiltPropMatch	Property match filtering	[RFCyyyy]
7	psampFiltHash	Hash filtering	[RFCyyyy]

The prerequisites set forth for addition of these OIDs are to be verified based on the content of this document.

Editor’s Note (to be removed prior to publication): the IANA is requested to assign a value for "xxx" under the ’mib-2’ subtree and to record the assignment in the SMI Numbers registry. When the assignment has been made, the RFC Editor is asked to replace "xxx" (here and in the MIB module) with the assigned value and to remove this note. The RFC editor is also asked to replace "yyyy" in this document and the MIB module by the number of the RFC when the assignment has been made.

9. Acknowledgment

This document is a product of the PSAMP and IPFIX working groups. The authors would like to thank the following persons: Paul Aitken for his detailed review, Dan Romascanu and the MIB doctors, and many more, for the technical reviews and feedback.

10. References

10.1. Normative References

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

[RFC2578] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.

[RFC2579] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Textual Conventions for SMIv2", STD 58, RFC 2579, April 1999.

[RFC2580] McCloghrie, K., Perkins, D., and J. Schoenwaelder, "Conformance Statements for SMIv2", STD 58, RFC 2580, April 1999.

[RFC2564] Kalbfleisch, C., Krupczak, C., Presuhn, R., and J. Saperia, "Application Management MIB", RFC 2564, May 1999.

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

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

[I-D.dkcm-ipfix-rfc5815bis] Dietz, T., Kobayashi, A., Claise, B., and G. Muenz, "Definitions of Managed Objects for IP Flow Information Export", draft-dkcm-ipfix-rfc5815bis-00 (work in progress), October 2011.

[RFC6340] Presuhn, R., "Textual Conventions for the Representation of Floating-Point Numbers", RFC 6340, August 2011.

10.2. Informative References

[RFC3410] Case, J., Mundy, R., Partain, D., and B. Stewart, "Introduction and Applicability Statements for Internet-Standard Management Framework", RFC 3410, December 2002.

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

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

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

Authors' Addresses

Thomas Dietz (editor)
NEC Europe Ltd.
NEC Laboratories Europe
Kurfuersten-Anlage 36
Heidelberg 69115
DE

Phone: +49 6221 4342-128
Email: dietz@neclab.eu

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

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

Juergen Quittek
NEC Europe Ltd.
NEC Laboratories Europe
Kurfuersten-Anlage 36
Heidelberg 69115
DE

Phone: +49 6221 4342-115
Email: quittek@neclab.eu

IPFIX Working Group T. Dietz, Ed.
Internet-Draft NEC Europe, Ltd.
Intended status: Standards Track A. Kobayashi
Expires: May 17, 2012 NTT PF Labs.
B. Claise
Cisco Systems, Inc.
G. Muenz
Technische Universitaet Muenchen
November 14, 2011

Definitions of Managed Objects for IP Flow Information Export
draft-ietf-ipfix-rfc5815bis-00.txt

Abstract

This document defines managed objects for IP Flow Information eXport (IPFIX). These objects provide information for monitoring IPFIX Exporters and IPFIX Collectors including the basic configuration information.

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 17, 2012.

Copyright Notice

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

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

Table of Contents

1. Introduction

This document defines two MIB modules for monitoring IP Flow Information eXport (IPFIX) Devices including Exporters and Collectors. Most of the objects defined by the IPFIX MIB module MUST be implemented. Some objects MAY be implemented corresponding to the functionality implemented in the equipment. Since the IPFIX architecture [RFC5470] foresees the possibility of using Filtering and/or Sampling functions to reduce the data volume, this document also provides the IPFIX SELECTOR MIB module, which contains the standardized selection methods and is controlled by IANA. The full configuration of the IPFIX Metering Process is out of the scope of these MIB modules.

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

2. IPFIX Documents Overview

The IPFIX protocol provides network administrators with access to IP Flow information. The architecture for the export of measured IP Flow information out of an IPFIX Exporting Process to a Collecting Process is defined in [RFC5470], per the requirements defined in [RFC3917]. The protocol document [RFC5101] specifies how IPFIX Data Records and Templates are carried via a congestion-aware transport protocol from IPFIX Exporting Processes to IPFIX Collecting Processes. IPFIX has a formal description of IPFIX Information Elements, their name, type and additional semantic information, as specified in [RFC5102]. 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.

It is assumed that Flow metering, export, and collection is performed according to the IPFIX architecture defined in [RFC5470]. The monitored configuration parameters of the export and collection of Flow Templates and Data Records is modeled according to [RFC5101]. Packet selection methods that may be optionally used by the IPFIX Metering Process are not considered in this MIB module. They are defined in the Packet Sampling (PSAMP) framework [RFC5474] and Sampling techniques [RFC5475] documents. Nevertheless, the basis for defining Sampling and Filtering functions is given with the IPFIX SELECTOR MIB module. Since the PSAMP export protocol [RFC5476] is based on the IPFIX protocol, the Sampling and Filtering functions can be added to the IPFIX SELECTOR MIB module as needed.

3. The Internet-Standard Management Framework

For a detailed overview of the documents that describe the current Internet-Standard Management Framework, please refer to section 7 of RFC 3410 [RFC3410].

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. MIB objects are generally accessed through the Simple Network Management Protocol (SNMP). Objects in the MIB are defined using the mechanisms defined in the Structure of Management Information (SMI). This memo specifies MIB modules that are compliant to the SMIv2, which is described in STD 58, RFC 2578 [RFC2578], STD 58, RFC 2579 [RFC2579] and STD 58, RFC 2580 [RFC2580].

4. Terminology

The definitions of the basic terms like IP Traffic Flow, Exporting Process, Collecting Process, Observation Points, etc. can be found in the IPFIX protocol document [RFC5101].

5. Structure of the IPFIX MIB

The IPFIX MIB module consists of seven main tables, the Transport Session table, the Template table and the corresponding Template Definition table, the Export table, the Metering Process table, the Observation Point table, and the Selection Process table. Since the IPFIX architecture [RFC5470] foresees the possibility of using Filtering and/or Sampling functions to reduce the data volume, the MIB module provides the basic objects for these functions with the Selection Process table. The IPFIX SELECTOR MIB module defined in the next section provides the standard Filtering and Sampling functions that can be referenced in the ipfixSelectionProcessTable.

All remaining objects contain statistical values for the different tables contained in the MIB module.

The following subsections describe all tables in the IPFIX MIB module.

5.1. The Transport Session Table

The Transport Session is the basis of the MIB module. The Transport Session table (ipfixTransportSessionTable) contains all Transport Sessions between Exporter and Collector. The table specifies the transport layer protocol of the Transport Session and, depending on that protocol, further parameters for the Transport Session. In the case of UDP and TCP, these are the source and destination address as well as the source and destination port. For Stream Control Transmission Protocol (SCTP), the table contains the SCTP Assoc Id, which is the index for the SCTP association in the SCTP MIB module [RFC3873]. The mode of operation of the device, i.e., if the Transport Session is used for collecting or exporting is given in the ipfixTransportSessionDeviceMode object. Further on, it contains the configured refresh parameters for Templates and Options Templates that are used across unreliable connections as UDP. Finally, the IPFIX version that is exported or collected by this Transport Session and a status of the Transport Session is given in the table.

To illustrate the use of the above tables, let us assume the following scenario: we have an Exporter on IP address 192.0.2.22 and a Collector on IP address 192.0.2.37. The Exporter uses TCP to export Templates and Data Records. The same Exporter also exports, with UDP, to a Collector with the IP address of 192.0.2.44. This would lead to the following Transport Session table on the Exporter:

```
    ipfixTransportSessionTable (1)
    |
    +- ipfixTransportSessionEntry (1)
       |
       +- index (5) (ipfixTransportSessionIndex)
       |  +- ipfixTransportSessionIndex (1) = 5
       |  +- ipfixTransportSessionProtocol (2) = 6 (TCP)
       |  +- ipfixTransportSessionSourceAddressType (3) = 1 (ipv4)
       |  +- ipfixTransportSessionSourceAddress (4) = 192.0.2.22
       |  +- ipfixTransportSessionDestinationAddressType (5) = 1 (ipv4)
       |  +- ipfixTransportSessionDestinationAddress (6) = 192.0.2.37
       |  +- ipfixTransportSessionSourcePort (7) = 7653
       |  +- ipfixTransportSessionDestinationPort (8) = 4739
       |  +- ipfixTransportSessionSctpAssocId (9) = 0
       |  +- ipfixTransportSessionDeviceMode (10) = exporting(1)
       |  +- ipfixTransportSessionTemplateRefreshTimeout (11) = 0
       |  +- ipfixTransportSessionOptionTemplateRefreshTimeout (12) = 0
       |  +- ipfixTransportSessionTemplateRefreshPacket (13) = 0
       |  +- ipfixTransportSessionOptionTemplateRefreshPacket (14) = 0
       |  +- ipfixTransportSessionIpfixVersion (15) = 10
       |  +- ipfixTransportSessionStatus (16) = 2 (active)
       .
       .
       .
       +- index (11) (ipfixTransportSessionIndex)
          +- ipfixTransportSessionIndex (1) = 11
          +- ipfixTransportSessionProtocol (2) = 17 (UDP)
          +- ipfixTransportSessionSourceAddressType (3) = 1 (ipv4)
          +- ipfixTransportSessionSourceAddress (4) = 192.0.2.22
          +- ipfixTransportSessionDestinationAddressType (5) = 1 (ipv4)
          +- ipfixTransportSessionDestinationAddress (6) = 192.0.2.44
          +- ipfixTransportSessionSourcePort (7) = 14287
          +- ipfixTransportSessionDestinationPort (8) = 4739
          +- ipfixTransportSessionSctpAssocId (9) = 0
          +- ipfixTransportSessionDeviceMode (10) = exporting(1)
          +- ipfixTransportSessionTemplateRefreshTimeout (11) = 100
          +- ipfixTransportSessionOptionTemplateRefreshTimeout (12)
          |                                                      = 100
          +- ipfixTransportSessionTemplateRefreshPacket (13) = 10
          +- ipfixTransportSessionOptionTemplateRefreshPacket (14) = 10
          +- ipfixTransportSessionIpfixVersion (15) = 10
          +- ipfixTransportSessionStatus (16) = 2 (active)
```

The values in brackets are the OID numbers. The Collectors would then have the same entry except that the index would most likely differ and the ipfixTransportSessionDeviceMode would be collecting(2).

5.2. The Template Table

The Template table lists all Templates (including Options Templates) that are sent (by an Exporter) or received (by a Collector). The (Options) Templates are unique per Transport Session, which also gives the device mode (Exporter or Collector) and Observation Domain; thus, the table is indexed by:

- the Transport Session Index (ipfixTransportSessionIndex)
- and the Observation Domain Id (ipfixTemplateObservationDomainId).

It contains the Set Id and an access time denoting the time when the (Options) Template was last sent or received.

To resume the above example, the Exporter may want to export a Template and an Options Template for each Transport Session defined above. This leads to the following Template table defining Template and Options Template:

```
 ipfixTemplateTable (3)
 |
 +- ipfixTemplateEntry (1)
    |
    +- index (5) (ipfixTransportSessionIndex)
    |  +- index (3) (ipfixTemplateObservationDomainId)
    |     + index (257) (ipfixTemplateId)
    |     | +- ipfixTemplateObservationDomainId (1) = 3
    |     | +- ipfixTemplateId (2) = 257
    |     | +- ipfixTemplateSetId (3) = 2
    |     | +- ipfixTemplateAccessTime (4)
    |     |                              = 2008-7-1,12:49:11.2,+2:0
    |     |
    |     + index (264) (ipfixTemplateId)
    |       +- ipfixTemplateObservationDomainId (1) = 3
    |       +- ipfixTemplateId (2) = 264
    |       +- ipfixTemplateSetId (3) = 3
    |       +- ipfixTemplateAccessTime (4)
    .                                = 2008-7-1,12:47:04.8,+2:0
    .
    .
    .
    +- index (11) (ipfixTransportSessionIndex)
       +- index (3) (ipfixTemplateObservationDomainId)
          + index (273) (ipfixTemplateId)
          | +- ipfixTemplateObservationDomainId (1) = 3
          | +- ipfixTemplateId (2) = 273
          | +- ipfixTemplateSetId (3) = 2
          | +- ipfixTemplateAccessTime (4)
          |                              = 2008-7-1,12:49:11.2,+2:0
          |
          + index (289) (ipfixTemplateId)
            +- ipfixTemplateObservationDomainId (1) = 3
            +- ipfixTemplateId (2) = 289
            +- ipfixTemplateSetId (3) = 3
            +- ipfixTemplateAccessTime (4)
                                         = 2008-7-1,12:47:04.8,+2:0
```

We assume that the Transport Session that is stored with index 5 in the Transport Session table of the Exporter is stored with index 17 in the Transport Session table of the (corresponding) Collector. Then, the Template table would look as follows:

```
ipfixTemplateTable (3)
|
+- ipfixTemplateEntry (1)
   |
   +- index (17) (ipfixTransportSessionIndex)
      +- index (3) (ipfixTemplateObservationDomainId)
         + index (257) (ipfixTemplateId)
         | +- ipfixTemplateObservationDomainId (1) = 3
         | +- ipfixTemplateId (2) = 257
         | +- ipfixTemplateSetId (3) = 2
         | +- ipfixTemplateAccessTime (4)
         |                            = 2008-7-1,12:49:11.8,+2:0
         |
         + index (264) (ipfixTemplateId)
           +- ipfixTemplateObservationDomainId (1) = 3
           +- ipfixTemplateId (2) = 264
           +- ipfixTemplateSetId (3) = 3
           +- ipfixTemplateAccessTime (4)
                                      = 2008-7-1,12:47:05.3,+2:0
```

The table on the second Collector would be analogous to the one shown above.

5.3. The Template Definition Table

The Template Definition table lists all the Information Elements contained in a Template or Options Template. Therefore, it has the same indexes as the corresponding Template table plus the Template Id. Its own index denotes the order of the Information Element inside the Template. Besides the Information Element Id and the length of the encoded value, the table contains the enterprise number for enterprise-specific Information Elements and flags for each Information Element. The flags indicate if the Information Element is used for scoping or as a Flow Key.

To resume the above example again, the Exporter is configured to export the octets received and dropped at the Observation Point since the last export of these values. In addition, it exports the start and end time of the Flow relative to the timestamp contained in the IPFIX header. This leads to the following Template Definition table on the Exporter:

```
    ipfixTemplateDefinitionTable (4)
    |
    +- ipfixTemplateDefinitionEntry (1)
       |
       +- index (5) (ipfixTransportSessionIndex)
          +- index (3) (ipfixTemplateObservationDomainId)
             + index (257) (ipfixTemplateId)
               +- index (1) (ipfixTemplateDefinitionIndex)
               |  +- ipfixTemplateDefinitionIndex (1) = 1
               |  +- ipfixTemplateDefinitionIeId (2) = 158
               |  |                      (flowStartDeltaMicroseconds)
               |  +- ipfixTemplateDefinitionIeLength (3) = 4
               |  +- ipfixTemplateDefinitionEnterprise (4) = 0
               |  +- ipfixTemplateDefinitionFlags (5) = 0
               |
               +- index (2) (ipfixTemplateDefinitionIndex)
               |  +- ipfixTemplateDefinitionIndex (1) = 2
               |  +- ipfixTemplateDefinitionIeId (2) = 159
               |  |                      (flowEndDeltaMicroseconds)
               |  +- ipfixTemplateDefinitionIeLength (3) = 4
               |  +- ipfixTemplateDefinitionEnterprise (4) = 0
               |  +- ipfixTemplateDefinitionFlags (5) = 0
               |
               +- index (3) (ipfixTemplateDefinitionIndex)
               |  +- ipfixTemplateDefinitionIndex (1) = 3
               |  +- ipfixTemplateDefinitionIeId (2) = 1
               |  |                                 (octetDeltaCount)
               |  +- ipfixTemplateDefinitionIeLength (3) = 8
               |  +- ipfixTemplateDefinitionEnterprise (4) = 0
               |  +- ipfixTemplateDefinitionFlags (5) = 0
               |
               +- index (4) (ipfixTemplateDefinitionIndex)
                  +- ipfixTemplateDefinitionIndex (1) = 4
                  +- ipfixTemplateDefinitionIeId (2) = 132
                  |                          (droppedOctetDeltaCount)
                  +- ipfixTemplateDefinitionIeLength (3) = 8
                  +- ipfixTemplateDefinitionEnterprise (4) = 0
                  +- ipfixTemplateDefinitionFlags (5) = 0
```

The corresponding table entry on the Collector is the same except that it would have another ipfixTransportSessionIndex, e.g., 17 as in the previous example.

5.4. The Export Table

On Exporters, the Export table (ipfixExportTable) can be used to support features like failover, load-balancing, duplicate export to several Collectors, etc. The table has three indexes that link an entry with:

o the Metering Process table (ipfixMeteringProcessCacheId, see below)

o and the Transport Session table (ipfixTransportSessionIndex).

Those entries with the same ipfixExportIndex and the same ipfixMeteringProcessCacheId define a Transport Session group. The member type for each group member describes its functionality. All Transport Sessions referenced in this table MUST have the ipfixTransportSessionDeviceMode exporting(1).

If the Exporter does not use Transport Session grouping, then each ipfixExportIndex contains a single ipfixMeteringProcessCacheId, and thus a singe Transport Session (ipfixTransportSessionIndex) and this session MUST have the member type primary(1).

For failover, a Transport Session group can contain one Transport Session with member type "primary" and several Transport Sessions with type secondary(2). Entries with other member types are not allowed for that type of group. For load-balancing or parallel export, all Transport Sessions in the group MUST have the same member type, either loadBalancing(4) or parallel(3).

The algorithms used for failover or load-balancing are out of the scope of this document.

To continue the example, we assume that the Exporter uses the two connections shown in the examples above as one primary Transport Session protected by a secondary Transport Session. The Exporter then has the following entries in the ipfixExportTable:

```
    ipfixExportTable (5)
    |
    +- ipfixExportEntry (1)
       |
       +- index (7) (ipfixExportIndex)
       |  +- index (9) (ipfixMeteringProcessCacheId)
       |     |  +- index (5) (ipfixTransportSessionIndex)
       |        |  +- ipfixExportIndex (1) = 7
       |        |  +- ipfixExportMemberType (2) = 1 (primary)
       |        |
       |        +- index (11) (ipfixTransportSessionIndex)
       |           +- ipfixExportIndex (1) = 7
       |           +- ipfixExportMemberType (2) = 2 (secondary)
       |
       +- index (8) (ipfixExportIndex)
          +- index (9) (ipfixMeteringProcessCacheId)
             +- index (5) (ipfixTransportSessionIndex)
             |  +- ipfixExportIndex (1) = 8
             |  +- ipfixExportMemberType (2) = 2 (secondary)
             +- index (11) (ipfixTransportSessionIndex)
                +- ipfixExportIndex (1) = 8
                +- ipfixExportMemberType (2) = 1 (primary)
```

The example shows that the Exporter uses the Metering Process Cache 9, explained below, to export IPFIX Data Records for the Transport Sessions 5 and 11. The Templates 257 and 264 defined above are exported within Transport Session 5, and the Templates 273 and 289 are exported within Transport Session 11. If we assume that Templates 257 and 264 are identical, then the Collector that receives Transport Session 11 is a backup for the Collector of Transport Session 5.

5.5. The Metering Process Table

The Metering Process, as defined in [RFC5101], consists of a set of functions. Maintaining the Flow Records is one of them. This function is responsible for passing the Flow Records to the Exporting Process and also for detecting Flow expiration. The Flow Records that are maintained by the Metering Process can be grouped by the Observation Points at which they are observed. The instance that maintains such a group of Flow Records is a kind of cache. For this reason, the Metering Process table (ipfixMeteringProcessTable) is indexed by cache Ids (ipfixMeteringProcessCacheId). Each cache can be maintained by a separate instance of the Metering Process. To specify the Observation Point(s) where the Flow Records are gathered, the ipfixMeteringProcessObservationPointGroupRef may contain an ipfixObservationPointGroupId from the Observation Point table (ipfixObservationPointTable) described in the next section. If an

Observation Point is not specified for the Flow Records, the ipfixMeteringProcessObservationPointGroupRef MUST be zero(0). The timeouts (ipfixMeteringProcessCacheActiveTimeout and ipfixMeteringProcessCacheInactiveTimeout) specify when Flows are expired.

```
ipfixMeteringProcessTable (6)
|
+- ipfixMeteringProcessEntry (1)
   |
   +- index (9) (ipfixMeteringProcessCacheId)
      +- ipfixMeteringProcessCacheId (1) = 9
      +- ipfixMeteringProcessObservationPointGroupRef (2) = 17
      +- ipfixMeteringProcessCacheActiveTimeout (3) = 100
      +- ipfixMeteringProcessCacheInactiveTimeout (4) = 100
```

5.6. The Observation Point Table

The Observation Point table (ipfixObservationPointTable) groups Observation Points with the ipfixObservationPointGroupId. Each entry contains the Observation Domain Id in which the Observation Point is located and a reference to the ENTITY MIB module [RFC4133] or the IF MIB module [RFC2863]. The objects in the ENTITY MIB module referenced by ipfixObservationPointPhysicalEntity or IF MIB module referenced by ipfixObservationPointPhysicalInterface denote the Observation Point. If no such index can be given in those modules, the references MUST be 0. If a reference is given in both object ipfixObservationPointPhysicalEntity and ipfixObservationPointPhysicalInterface, then both MUST point to the same physical interface. In addition, a direction can be given to render more specifically which Flow to monitor.

```
    ipfixObservationPointTable (7)
    |
    +- ipfixObservationPointEntry (1)
       |
       +- index (17) (ipfixObservationPointGroupId)
          +- index (1) (ipfixObservationPointIndex)
          |  +- ipfixObservationPointGroupId (1) = 17
          |  +- ipfixObservationPointIndex (2) = 1
          |  +- ipfixObservationPointObservationDomainId (3) = 3
          |  +- ipfixObservationPointPhysicalEntity (4) = 6
          |  +- ipfixObservationPointPhysicalInterface(5) = 0
          |  +- ipfixObservationPointPhysicalEntityDirection (6)
                                                     = 3 (both)
          |
          +- index (2) (ipfixObservationPointIndex)
             +- ipfixObservationPointGroupId (1) = 17
             +- ipfixObservationPointIndex (2) = 2
             +- ipfixObservationPointObservationDomainId (3) = 3
             +- ipfixObservationPointPhysicalEntity (4) = 0
             +- ipfixObservationPointPhysicalInterface (5) = 0
             +- ipfixObservationPointPhysicalEntityDirection (6)
                                                   = 1 (ingress)
```

5.7. The Selection Process Table

This table supports the usage of Filtering and Sampling functions, as described in [RFC5470]. It contains lists of functions per Metering Process cache (ipfixMeteringProcessCacheId). The selection process index ipfixSelectionProcessIndex forms groups of selection methods that are applied to an observed packet stream. The selection process selector index (ipfixSelectionProcessSelectorIndex) indicates the order in which the functions are applied to the packets observed at the Observation Points associated with the Metering Process cache. The selection methods are applied in increasing order, i.e., selection methods with a lower ipfixSelectionProcessSelectorIndex are applied first. The functions are referred by object identifiers pointing to the function with its parameters. If the selection method does not use parameters, then it MUST point to the root of the function subtree (see also Section 6). If the function uses parameters, then it MUST point to an entry in the parameter table of the selection method. If no Filtering or Sampling function is used for a Metering Process, then an entry for the Metering Process SHOULD be created pointing to the Select All function (ipfixFuncSelectAll).

5.8. The Statistical Tables

For the ipfixTransportSessionTable, the ipfixTemplateTable, the ipfixMeteringProcessTable, and the ipfixSelectionProcessTable statistical tables are defined that augment those tables. All the statistical tables contain a discontinuity object that holds a timestamp that denotes the time when a discontinuity event occurred to notify the management system that the counters contained in those tables might not be continuous anymore.

5.8.1. The Transport Session Statistical Table

The Transport Session Statistical table (ipfixTransportSessionStatsTable) augments the ipfixTransportSessionTable with statistical values. It contains the rate (in bytes per second) with which it receives or sends out IPFIX Messages, the number of bytes, packets, messages, Records, Templates and Options Templates received or sent and the number of messages that were discarded.

5.8.2. The Template Statistical Table

This table contains a statistical value for each Template. It augments the Template table (ipfixTemplateTable) and specifies the number of Data Records exported or collected for the Template.

5.8.3. The Metering Process Statistical Table

This table augments the Metering Process table (ipfixMeteringProcessTable). It contains the statistical values for the exported Data Records and the number of unused cache entries.

5.8.4. The Selection Process Statistical Table

This table augments the Selection Process table (ipfixSelectionProcessTable) and introduces two generic statistical values, the number of packets observed and the number of packets dropped by the selection method.

6. Structure of the IPFIX SELECTOR MIB

The IPFIX SELECTOR MIB module defined in this section provides the standard Filtering and Sampling functions that can be referenced in the ipfixSelectionProcessTable. All standard Filtering and Sampling functions MUST be registered in the subtree under object ipfixSelectorFunctions (iso.org.dod.internet.mgmt.mib-2.ipfixSelectorMIB, or as numbers 1.3.6.1.2.1.194). The toplevel OIDs in the subtree under object ipfixSelectorFunctions MUST be registered in a subregistry maintained by IANA at http://www.iana.org/assignments/smi-numbers. The first entry in this subtree is the Select All function (ipfixFuncSelectAll) defined in this document as { ipfixSelectorFunctions 1}. Further selector functions MUST be registered at IANA and are subject to Expert Review [RFC5226], i.e., review by one of a group of experts designated by an IETF Area Director. The group of experts MUST check the requested MIB objects for completeness and accuracy of the description. Requests for MIB objects that duplicate the functionality of existing objects SHOULD be declined. The smallest available OID SHOULD be assigned to a new MIB objects. The specification of new MIB objects SHOULD follow the structure specified in the next Section and MUST be published using a well-established and persistent publication medium. The experts will initially be drawn from the Working Group Chairs and document editors of the IPFIX and PSAMP Working Groups.

6.1. The Selector Functions

The following figure shows what the MIB tree usually should look like. It already contains the ipfixFuncSelectAll. The subtree in ipfixFuncF2 gives the basic structure that all selection methods SHOULD follow.

```
ipfixSelectorFunctions
|
+- ipfixFuncSelectAll
|  |
|  +- ipfixFuncSelectAllAvail (is the function available?)
|
+- ipfixFuncF2
|  |
|  +- ipfixFuncF2Avail (is the function F2 available?)
|  |
|  +- ipfixFuncF2Parameters (a table with parameters)
...
|
+- ipfixFunFn...
```

The selection method SHOULD be designed as a MIB subtree introduced by an object with the name ipfixFunc appended by a function name. The objects in this subtree SHOULD be prefixed by this name. If the function is named Fx, then we would start a subtree with an OID named ipfixFuncFx. This subtree should contain an object ipfixFuncFxAvail that has the type TruthValue. If a selection method takes parameters, the MIB should contain a table named ipfixFuncFxParameters, which should contain all the parameters that the selection method specifies. An entry in this table will be referenced by the IPFIX MIB module if the selection method with the parameters is used.

To illustrate the structure defined above, the following contains an example of a function MyFunc that holds three integer parameters Param1, Param2, and Param3. In the example, there are currently two instances of the parameters set defined with indexes 1 and 4.

```
 ipfixSelectorFunctions (1)
 |
 +- ipfixFuncMyFunc (?)
    |
    +- ipfixFuncMyFuncAvail (1) = true
    +- ipfixFuncMyFuncParameters (2)
       |
       +- ipfixFuncMyFuncParametersEntry (1)
          |
          +- index (1) (ipfixFuncMyFuncParametersIndex)
          |  +- ipfixFuncMyFuncParam1 (1) = 47
          |  +- ipfixFuncMyFuncParam2 (2) = -128
          |  +- ipficFuncMyFuncParam3 (3) = 19
          |
          +- index(4) (ipfixFuncMyFuncParametersIndex)
             +- ipfixFuncMyFuncParam1 (1) = 19
             +- ipfixFuncMyFuncParam2 (2) = -1
             +- ipficFuncMyFuncParam3 (3) = 728
```

If the function defined above is referenced in the IPFIX MIB module, the ipfixSelectionProcessTable would look as follows:

```
ipfixSelectionProcessTable (8)
|
+- ipfixSelectionProcessEntry (1)
   |
   +- index (9) (ipfixMeteringProcessCacheId)
      +- index (1) (ipfixSelectionProcessIndex)
         +- index (1) (ipfixSelectionProcessSelectorIndex)
         |  +- ipfixSelectionProcessSelectorFunction (3)
         |                           = ipfixSelectorFunctions.?.2.1.4
         +- index (2) (ipfixSelectionProcessSelectorIndex)
            +- ipfixSelectionProcessSelectorFunction (3)
                                     = ipfixSelectorFunctions.?.2.1.1
```

This means that for the ipfixMeteringProcessCacheId(9), a Selection Process with index 1 is created that applies two times the same function but with different parameter sets. First, the function MyFunc is applied with the parameters of the set with index 4 and the with the parameters of the set with index 1.

7. Relationship to Other MIB Modules

Besides the usual imports from the SNMP Standards [RFC2578], [RFC2579], and [RFC2580], the IPFIX MIB module references the ENTITY MIB module [RFC4133] and the IF MIB module [RFC2863].

7.1. Relationship to the ENTITY MIB and IF MIB

The Observation Point table (ipfixObservationPointTable) contains a reference to the ENTITY MIB module[RFC4133] (ipfixObservationPointPhysicalEntity) or the IF MIB module [RFC2863] (ipfixObservationPointPhysicalInterface). If the implementors of the IPFIX MIB module want to specify the physical entity where Flows are observed, then they SHOULD also implement the ENTITY MIB and/or the IF MIB module. The implementation of the ENTITY MIB and/or IF MIB module is OPTIONAL. If one of them is not implemented, then all values of the respective column ipfixObservationPointPhysicalEntity or ipfixObservationPointPhysicalInterface in the Observation Point table are zero and the values of the ipfixObservationPointPhysicalEntityDirection columns are unknown(0), if none of them are defined.

7.2. MIB Modules Required for IMPORTS

The IPFIX MIB module requires the modules SNMPv2-SMI [RFC2578], SNMPv2-TC [RFC2579], and SNMPv2-CONF [RFC2580]. Further on, it imports the textual conventions InetAddressType and InetAddress from the INET ADDRESS MIB module [RFC4001].

The IPFIX SELECTOR MIB module also requires the modules SNMPv2-SMI [RFC2578], SNMPv2-TC [RFC2579], and SNMPv2-CONF [RFC2580].

8. MIB Definitions

This section contains the definitions of the IPFIX-MIB module and the IPFIX-SELECTOR-MIB module. There are different mandatory groups defined for Collector and Exporter implementations. The statistical objects are made OPTIONAL.

8.1. IPFIX MIB Definition

```
   IPFIX-MIB DEFINITIONS ::= BEGIN

   IMPORTS
       MODULE-IDENTITY, OBJECT-TYPE, mib-2, Unsigned32, Counter64,
       Gauge32
           FROM SNMPv2-SMI                               -- RFC2578
       TimeStamp, DateAndTime
           FROM SNMPv2-TC                                -- RFC2579
       MODULE-COMPLIANCE, OBJECT-GROUP
           FROM SNMPv2-CONF                              -- RFC2580
       InterfaceIndexOrZero
           FROM IF-MIB                                   -- RFC2863
       InetAddressType, InetAddress, InetPortNumber
           FROM INET-ADDRESS-MIB                         -- RFC4001
       PhysicalIndexOrZero
           FROM ENTITY-MIB;                              -- RFC4133

   ipfixMIB MODULE-IDENTITY
       LAST-UPDATED "201004190000Z"          -- 19 April 2010
       ORGANIZATION "IETF IPFIX Working Group"
       CONTACT-INFO
           "WG charter:
             http://www.ietf.org/html.charters/ipfix-charter.html

           Mailing Lists:
             General Discussion: ipfix@ietf.org
             To Subscribe: http://www1.ietf.org/mailman/listinfo/ipfix
             Archive:
         http://www1.ietf.org/mail-archive/web/ipfix/current/index.html

           Editor:
             Thomas Dietz
             NEC Europe Ltd.
             NEC Laboratories Europe
             Network Research Division
             Kurfuersten-Anlage 36
             69115 Heidelberg
             Germany
```

```
             Phone: +49 6221 4342-128
             Email: Thomas.Dietz@nw.neclab.eu

             Atsushi Kobayashi
             NTT Information Sharing Platform Laboratories
             3-9-11 Midori-cho
             Musashino-shi
             180-8585
             Japan
             Phone: +81-422-59-3978
             Email: akoba@nttv6.net

             Benoit Claise
             Cisco Systems, Inc.
             De Kleetlaan 6a b1
             Degem 1831
             Belgium
             Phone:  +32 2 704 5622
             Email: bclaise@cisco.com

             Gerhard Muenz
             Technische Universitaet Muenchen
             Department of Informatics
             Chair for Network Architectures and Services (I8)
             Boltzmannstr. 3
             85748 Garching
             Germany
             Phone: +49 89 289-18008
             Email: muenz@net.in.tum.de
             URI:   http://www.net.in.tum.de/~muenz"
       DESCRIPTION
           "The IPFIX MIB defines managed objects for IP Flow
           Information eXport.  These objects provide information about
           managed nodes supporting the IPFIX protocol,
           for Exporters as well as for Collectors.

           Copyright (c) 2010 IETF Trust and the persons identified as
           authors of the code.  All rights reserved.

           Redistribution and use in source and binary forms, with or
           without modification, is permitted pursuant to, and subject
           to the license terms contained in, the Simplified BSD
           License set forth in Section 4.c of the IETF Trust's
           Legal Provisions Relating to IETF Documents
           (http://trustee.ietf.org/license-info)."

   --  Revision history
```

```
        REVISION     "201004190000Z"         -- 19 April 2010
        DESCRIPTION
           "Initial version, published as RFC 5815."

       ::= { mib-2 193 }

   --******************************************************************
   -- Top Level Structure of the MIB
   --******************************************************************

   ipfixObjects     OBJECT IDENTIFIER ::= { ipfixMIB 1 }
   ipfixConformance OBJECT IDENTIFIER ::= { ipfixMIB 2 }

   ipfixMainObjects OBJECT IDENTIFIER ::= { ipfixObjects 1 }
   ipfixStatistics  OBJECT IDENTIFIER ::= { ipfixObjects 2 }

   --==================================================================
   -- 1.1: Objects used by all IPFIX implementations
   --==================================================================
   --------------------------------------------------------------------
   -- 1.1.1: Transport Session Table
   --------------------------------------------------------------------
   ipfixTransportSessionTable  OBJECT-TYPE
       SYNTAX      SEQUENCE OF IpfixTransportSessionEntry
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "This table lists the currently established Transport
           Sessions between an Exporting Process and a Collecting
           Process."
       ::= { ipfixMainObjects 1 }

   ipfixTransportSessionEntry OBJECT-TYPE
       SYNTAX      IpfixTransportSessionEntry
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "Defines an entry in the ipfixTransportSessionTable."
       INDEX       { ipfixTransportSessionIndex }
       ::= { ipfixTransportSessionTable 1 }

   IpfixTransportSessionEntry ::=
       SEQUENCE {
          ipfixTransportSessionIndex                 Unsigned32,
          ipfixTransportSessionProtocol              Unsigned32,
          ipfixTransportSessionSourceAddressType     InetAddressType,
          ipfixTransportSessionSourceAddress         InetAddress,
          ipfixTransportSessionDestinationAddressType InetAddressType,
```

```
          ipfixTransportSessionDestinationAddress     InetAddress,
          ipfixTransportSessionSourcePort             InetPortNumber,
          ipfixTransportSessionDestinationPort        InetPortNumber,
          ipfixTransportSessionSctpAssocId            Unsigned32,
          ipfixTransportSessionDeviceMode             INTEGER,
          ipfixTransportSessionTemplateRefreshTimeout Unsigned32,
          ipfixTransportSessionOptionsTemplateRefreshTimeout Unsigned32,
          ipfixTransportSessionTemplateRefreshPacket  Unsigned32,
          ipfixTransportSessionOptionsTemplateRefreshPacket Unsigned32,
          ipfixTransportSessionIpfixVersion           Unsigned32,
          ipfixTransportSessionStatus                 INTEGER
      }

   ipfixTransportSessionIndex OBJECT-TYPE
       SYNTAX      Unsigned32 (1..4294967295)
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "Locally arbitrary, but unique identifier of an entry in
           the ipfixTransportSessionTable.  The value is expected to
           remain constant from a re-initialization of the entity's
           network management agent to the next re-initialization."
       ::= { ipfixTransportSessionEntry 1 }

   ipfixTransportSessionProtocol OBJECT-TYPE
       SYNTAX      Unsigned32 (1..255)
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "The transport protocol used for receiving or transmitting
           IPFIX Messages.  Protocol numbers are assigned by IANA.  A
           current list of all assignments is available from
           <http://www.iana.org/>."
       REFERENCE
           "RFC 5101, Specification of the IP Flow
           Information Export (IPFIX) Protocol for the Exchange of IP
           Traffic Flow Information, Section 10."
       ::= { ipfixTransportSessionEntry 2 }

   ipfixTransportSessionSourceAddressType OBJECT-TYPE
       SYNTAX      InetAddressType
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "The type of address used for the source address,
           as specified in RFC 4001.  This object is used with protocols
           (specified in ipfixTransportSessionProtocol) like TCP (6)
           and UDP (17) that have the notion of addresses.  SCTP (132)
```

```
          should use the ipfixTransportSessionSctpAssocId instead.
          If SCTP (132) or any other protocol without the notion of
          addresses is used, the object MUST be set to unknown(0)."
      ::= { ipfixTransportSessionEntry 3 }

   ipfixTransportSessionSourceAddress OBJECT-TYPE
      SYNTAX      InetAddress
      MAX-ACCESS  read-only
      STATUS      current
      DESCRIPTION
          "The source address of the Exporter of the IPFIX Transport
          Session.  This value is interpreted according to the value of
          ipfixTransportSessionAddressType as specified in RFC 4001.
          This object is used with protocols (specified in
          ipfixTransportSessionProtocol) like TCP (6) and UDP (17) that
          have the notion of addresses.  SCTP (132) should use the
          ipfixTransportSessionSctpAssocId instead.  If SCTP (132) or
          any other protocol without the notion of addresses is used,
          the object MUST be set to a zero-length string."
      ::= { ipfixTransportSessionEntry 4 }

   ipfixTransportSessionDestinationAddressType OBJECT-TYPE
      SYNTAX      InetAddressType
      MAX-ACCESS  read-only
      STATUS      current
      DESCRIPTION
          "The type of address used for the destination address,
          as specified in RFC 4001.  This object is used with protocols
          (specified in ipfixTransportSessionProtocol) like TCP (6)
          and UDP (17) that have the notion of addresses.  SCTP (132)
          should use the ipfixTransportSessionSctpAssocId instead.
          If SCTP (132) or any other protocol without the notion of
          addresses is used, the object MUST be set to unknown(0)."
      ::= { ipfixTransportSessionEntry 5 }

   ipfixTransportSessionDestinationAddress OBJECT-TYPE
      SYNTAX      InetAddress
      MAX-ACCESS  read-only
      STATUS      current

      DESCRIPTION
          "The destination address of the Collector of the IPFIX
          Transport Session.  This value is interpreted according to
          the value of ipfixTransportSessionAddressType, as specified
          in RFC 4001.  This object is used with protocols
          (specified in ipfixTransportSessionProtocol) like TCP (6)
          and UDP (17) that have the notion of addresses.  SCTP (132)
          should use the ipfixTransportSessionSctpAssocId instead.
```

```
            If SCTP (132) or any other protocol without the notion of
            addresses is used, the object MUST be set to a zero-length
            string"
        ::= { ipfixTransportSessionEntry 6 }

    ipfixTransportSessionSourcePort OBJECT-TYPE
        SYNTAX      InetPortNumber
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The transport protocol port number of the Exporter.
            This object is used with protocols (specified in
            ipfixTransportSessionProtocol) like TCP (6)
            and UDP (17) that have the notion of ports.  SCTP (132)
            should copy the value of sctpAssocLocalPort if the
            Transport Session is in collecting mode or
            sctpAssocRemPort if the Transport Session is in
            exporting mode.  The association is referenced
            by the ipfixTransportSessionSctpAssocId.
            If any other protocol without the notion of
            ports is used, the object MUST be set to zero."
        ::= { ipfixTransportSessionEntry 7 }

    ipfixTransportSessionDestinationPort OBJECT-TYPE
        SYNTAX      InetPortNumber
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The transport protocol port number of the Collector.  The
            default value is 4739 for all currently defined transport
            protocol types.  This object is used with protocols
            (specified in ipfixTransportSessionProtocol) like TCP (6)
            and UDP (17) that have the notion of ports.  SCTP (132)
            should copy the value of sctpAssocRemPort if the
            Transport Session is in collecting mode or
            sctpAssocLocalPort if the Transport Session is in
            exporting mode.  The association is referenced
            by the ipfixTransportSessionSctpAssocId.
            If any other protocol without the notion of
            ports is used, the object MUST be set to zero."

        ::= { ipfixTransportSessionEntry 8 }

    ipfixTransportSessionSctpAssocId OBJECT-TYPE
        SYNTAX      Unsigned32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
```

```
           "The association id used for the SCTP session between the
           Exporter and the Collector of the IPFIX Transport Session.
           It is equal to the sctpAssocId entry in the sctpAssocTable
           defined in the SCTP MIB.  This object is only valid if
           ipfixTransportSessionProtocol has the value 132 (SCTP).  In
           all other cases, the value MUST be zero."
       REFERENCE
           "RFC 3873, Stream Control Transmission Protocol (SCTP)
           Management Information Base (MIB)."
       ::= { ipfixTransportSessionEntry 9 }

   ipfixTransportSessionDeviceMode OBJECT-TYPE
       SYNTAX      INTEGER {
                       exporting(1),
                       collecting(2)
                   }
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "The mode of operation of the device for the given Transport
           Session.  This object can have the following values:

           exporting(1)
               This value MUST be used if the Transport Session is
               used for exporting Records to other IPFIX Devices,
               i.e., this device acts as Exporter.

           collecting(2)
               This value MUST be used if the Transport Session is
               used for collecting Records from other IPFIX Devices,
               i.e., this device acts as Collector."
       ::= { ipfixTransportSessionEntry 10 }

   ipfixTransportSessionTemplateRefreshTimeout OBJECT-TYPE
       SYNTAX      Unsigned32
       UNITS       "seconds"
       MAX-ACCESS  read-only
       STATUS      current

       DESCRIPTION
           "On Exporters, this object contains the time in seconds
           after which IPFIX Templates are resent by the
           Exporter.

           On Collectors, this object contains the lifetime in seconds
           after which a Template becomes invalid when it is not
           received again within this lifetime.
```

```
           This object is only valid if ipfixTransportSessionProtocol
           has the value 17 (UDP).  In all other cases, the value MUST
           be zero."
       REFERENCE
           "RFC 5101, Specification of the IP Flow Information Export
           (IPFIX) Protocol for the Exchange of IP Traffic Flow
           Information, Sections 10.3.6 and 10.3.7."
       ::= { ipfixTransportSessionEntry 11 }

   ipfixTransportSessionOptionsTemplateRefreshTimeout OBJECT-TYPE
       SYNTAX      Unsigned32
       UNITS       "seconds"
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "On Exporters, this object contains the time in seconds
           after which IPFIX Options Templates are resent by the
           Exporter.

           On Collectors, this object contains the lifetime in seconds
           after which an Options Template becomes invalid when it is
           not received again within this lifetime.

           This object is only valid if ipfixTransportSessionProtocol
           has the value 17 (UDP).  In all other cases the value MUST
           be zero."
       REFERENCE
           "RFC 5101, Specification of the IP Flow Information Export
           (IPFIX) Protocol for the Exchange of IP Traffic Flow
           Information, Sections 10.3.6 and 10.3.7."
       ::= { ipfixTransportSessionEntry 12 }

   ipfixTransportSessionTemplateRefreshPacket OBJECT-TYPE
       SYNTAX      Unsigned32
       UNITS       "packets"
       MAX-ACCESS  read-only
       STATUS      current

       DESCRIPTION
           "On Exporters, this object contains the number of exported
           IPFIX Messages after which IPFIX Templates are resent
           by the Exporter.

           On Collectors, this object contains the lifetime in number
           of exported IPFIX Messages after which a Template becomes
           invalid when it is not received again within this lifetime.

           This object is only valid if ipfixTransportSessionProtocol
```

```
           has the value 17 (UDP).  In all other cases the value MUST
           be zero."
       REFERENCE
           "RFC 5101, Specification of the IP Flow Information Export
           (IPFIX) Protocol for the Exchange of IP Traffic Flow
           Information, Sections 10.3.6 and 10.3.7."
       ::= { ipfixTransportSessionEntry 13 }

   ipfixTransportSessionOptionsTemplateRefreshPacket OBJECT-TYPE
       SYNTAX      Unsigned32
       UNITS       "packets"
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "On Exporters, this object contains the number of exported
           IPFIX Messages after which IPFIX Options Templates are
           resent by the Exporter.

           On Collectors, this object contains the lifetime in number
           of exported IPFIX Messages after which an Options Template
           becomes invalid when it is not received again within this
           lifetime.

           This object is only valid if ipfixTransportSessionProtocol
           has the value 17 (UDP).  In all other cases the value MUST
           be zero."
       REFERENCE
           "RFC 5101, Specification of the IP Flow Information Export
           (IPFIX) Protocol for the Exchange of IP Traffic Flow
           Information, Sections 10.3.6 and 10.3.7."
       ::= { ipfixTransportSessionEntry 14 }

   ipfixTransportSessionIpfixVersion OBJECT-TYPE
       SYNTAX      Unsigned32 (0..65535)
       MAX-ACCESS  read-only
       STATUS      current

       DESCRIPTION
           "On Exporters the object contains the version number of the
           IPFIX protocol that the Exporter uses to export its data in
           this Transport Session.

           On Collectors the object contains the version number of the
           IPFIX protocol it receives for this Transport Session.

           If IPFIX Messages of different IPFIX protocol versions are
           transmitted or received in this Transport Session, this
           object contains the maximum version number."
```

```
        REFERENCE
            "RFC 5101, Specification of the IP Flow Information Export
            (IPFIX) Protocol for the Exchange of IP Traffic Flow
            Information, Section 3.1."
        ::= { ipfixTransportSessionEntry 15 }

    ipfixTransportSessionStatus OBJECT-TYPE
        SYNTAX      INTEGER {
                        unknown(0),
                        inactive(1),
                        active(2)
                    }
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The status of a Transport Session.  This object can have the
            following values:

            unknown(0)
                This value MUST be used if the status of the
                Transport Session cannot be detected by the equipment.
                This value should be avoided as far as possible.

            inactive(1)
                This value MUST be used for Transport Sessions that
                are specified in the system but are not currently active.
                The value can be used, e.g., for Transport Sessions that
                are backup (secondary) sessions in a Transport Session
                group.

            active(2)
                This value MUST be used for Transport Sessions that are
                currently active and transmitting or receiving data."
        ::= { ipfixTransportSessionEntry 16 }

    --------------------------------------------------------------------
    -- 1.1.2: Template Table
    --------------------------------------------------------------------
    ipfixTemplateTable  OBJECT-TYPE
        SYNTAX      SEQUENCE OF IpfixTemplateEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "This table lists the Templates and Options Templates that
            are transmitted by the Exporting Process or received by the
            Collecting Process.

            The table contains the Templates and Options Templates that
```

```
            are received or used for exporting data for a given
            Transport Session group and Observation Domain.

            Withdrawn or invalidated (Options) Template MUST be removed
            from this table."
        ::= { ipfixMainObjects 2 }

    ipfixTemplateEntry OBJECT-TYPE
        SYNTAX      IpfixTemplateEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "Defines an entry in the ipfixTemplateTable."
        INDEX       {
            ipfixTransportSessionIndex,
            ipfixTemplateObservationDomainId,
            ipfixTemplateId
        }
        ::= { ipfixTemplateTable 1 }

    IpfixTemplateEntry ::=
        SEQUENCE {
            ipfixTemplateObservationDomainId Unsigned32,
            ipfixTemplateId                  Unsigned32,
            ipfixTemplateSetId               Unsigned32,
            ipfixTemplateAccessTime          DateAndTime
        }

    ipfixTemplateObservationDomainId OBJECT-TYPE
        SYNTAX      Unsigned32 (0..4294967295)
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "The Id of the Observation Domain for which this Template
            is defined.  This value is used when sending IPFIX Messages.

            The special value of 0 indicates that the Data Records
            exported with this (Option Template) cannot be applied to a
            single Observation Domain."
        REFERENCE
            "RFC 5101, Specification of the IP Flow Information Export
            (IPFIX) Protocol for the Exchange of IP Traffic Flow
            Information, Section 3.1."
        ::= { ipfixTemplateEntry 1 }

    ipfixTemplateId OBJECT-TYPE
        SYNTAX      Unsigned32 (256..65535)
        MAX-ACCESS  not-accessible
```

```
       STATUS      current
       DESCRIPTION
           "This number indicates the Template Id in the IPFIX
           Message.  Values from 0 to 255 are not allowed for Template
           Ids."
       REFERENCE
           "RFC 5101, Specification of the IP Flow Information Export
           (IPFIX) Protocol for the Exchange of IP Traffic Flow
           Information, Section 3.4.1."
       ::= { ipfixTemplateEntry 2 }

   ipfixTemplateSetId OBJECT-TYPE
       SYNTAX      Unsigned32 (1..65535)
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "This number indicates the Set Id of the Template.  This
           object allows to easily retrieve the Template type.

           Currently, there are two values defined.  The value 2 is
           used for Sets containing Template definitions.  The value 3
           is used for Sets containing Options Template definitions."
       REFERENCE
           "RFC 5101, Specification of the IP Flow Information Export
           (IPFIX) Protocol for the Exchange of IP Traffic Flow
           Information, Section 3.3.2."
       ::= { ipfixTemplateEntry 3 }

   ipfixTemplateAccessTime OBJECT-TYPE
       SYNTAX      DateAndTime
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "If the Transport Session is in exporting mode
           (ipfixTransportSessionDeviceMode) the time when this
           (Options) Template was last sent to the Collector(s).

           In the specific case of UDP as transport protocol, this
           time is used to know when a retransmission of the
           (Options) Template is needed.

           If it is in collecting mode, this object contains the
           time when this (Options) Template was last received from
           the Exporter.  In the specific case of UDP as transport
           protocol, this time is used to know when this (Options)
           Template times out and thus is no longer valid."
       ::= { ipfixTemplateEntry 4 }
```

```
   --------------------------------------------------------------------
   -- 1.1.3: Exported Template Definition Table
   --------------------------------------------------------------------
   ipfixTemplateDefinitionTable  OBJECT-TYPE
       SYNTAX      SEQUENCE OF IpfixTemplateDefinitionEntry
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "On Exporters, this table lists the (Options) Template fields
           of which a (Options) Template is defined.  It defines the
           (Options) Template given in the ipfixTemplateId specified in
           the ipfixTemplateTable.

           On Collectors, this table lists the (Options) Template fields
           of which a (Options) Template is defined.  It defines the
           (Options) Template given in the ipfixTemplateId specified in
           the ipfixTemplateTable."
       ::= { ipfixMainObjects 3 }

   ipfixTemplateDefinitionEntry OBJECT-TYPE
       SYNTAX      IpfixTemplateDefinitionEntry
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "Defines an entry in the ipfixTemplateDefinitionTable."

       INDEX       {
           ipfixTransportSessionIndex,
           ipfixTemplateObservationDomainId,
           ipfixTemplateId,
           ipfixTemplateDefinitionIndex
       }
       ::= { ipfixTemplateDefinitionTable 1 }

   IpfixTemplateDefinitionEntry ::=
       SEQUENCE {
           ipfixTemplateDefinitionIndex            Unsigned32,
           ipfixTemplateDefinitionIeId             Unsigned32,
           ipfixTemplateDefinitionIeLength         Unsigned32,
           ipfixTemplateDefinitionEnterpriseNumber Unsigned32,
           ipfixTemplateDefinitionFlags            BITS
       }

   ipfixTemplateDefinitionIndex OBJECT-TYPE
       SYNTAX      Unsigned32 (1..65535)
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
```

```
        "The ipfixTemplateDefinitionIndex specifies the order in
        which the Information Elements are used in the (Options)
        Template Record.

        Since a Template Record can contain a maximum of 65535
        Information Elements, the index is limited to this value."
    REFERENCE
        "RFC 5101, Specification of the IP Flow Information Export
        (IPFIX) Protocol for the Exchange of IP Traffic Flow
        Information, Sections 3.4.1 and 3.4.2."
    ::= { ipfixTemplateDefinitionEntry 1 }

ipfixTemplateDefinitionIeId OBJECT-TYPE
    SYNTAX      Unsigned32 (1..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This indicates the Information Element Id at position
        ipfixTemplateDefinitionIndex in the (Options) Template
        ipfixTemplateId.  This implicitly specifies the data type
        of the Information Element.  The elements are registered
        at IANA.  A current list of assignments can be found at
        <http://www.iana.org/assignments/ipfix>"

    REFERENCE
        "RFC 5101, Specification of the IP Flow Information Export
        (IPFIX) Protocol for the Exchange of IP Traffic Flow
        Information, Section 3.2.

        RFC 5102, Information Model for IP Flow Information Export."
    ::= { ipfixTemplateDefinitionEntry 2 }

ipfixTemplateDefinitionIeLength OBJECT-TYPE
    SYNTAX      Unsigned32 (0..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This indicates the length of the Information Element Id at
        position ipfixTemplateDefinitionIndex in the (Options)
        Template ipfixTemplateId."
    REFERENCE
        "RFC 5101, Specification of the IP Flow Information Export
        (IPFIX) Protocol for the Exchange of IP Traffic Flow
        Information, Section 3.2.

        RFC 5102, Information Model for IP Flow Information Export."
    ::= { ipfixTemplateDefinitionEntry 3 }
```

```
   ipfixTemplateDefinitionEnterpriseNumber OBJECT-TYPE
       SYNTAX      Unsigned32
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "IANA enterprise number of the authority defining the
           Information Element identifier in this Template Record.
           Enterprise numbers are assigned by IANA.  A current list of
           all assignments is available from
           <http://www.iana.org/assignments/enterprise-numbers/>.

           This object must be zero(0) for all standard Information
           Elements registered with IANA.  A current list of these
           elements is available from
           <http://www.iana.org/assignments/ipfix/ipfix.xhtml>."
       REFERENCE
           "RFC 5101, Specification of the IP Flow Information Export
           (IPFIX) Protocol for the Exchange of IP Traffic Flow
           Information, Section 3.2.

           RFC 5102, Information Model for IP Flow Information Export."
       ::= { ipfixTemplateDefinitionEntry 4 }

   ipfixTemplateDefinitionFlags OBJECT-TYPE
       SYNTAX      BITS {
                       scope(0),
                       flowKey(1)
                   }
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "This bitmask indicates special attributes for the
           Information Element:

           scope(0)
               This Information Element is used for scope.

           flowKey(1)
               This Information Element is a Flow Key.

           Thus, we get the following values for an Information Element:

           If neither bit scope(0) nor bit flowKey(1) are set
               The Information Element is neither used for scoping nor
               as Flow Key.
           If only bit scope(0) is set
               The Information Element is used for scoping.
           If only bit flowKey(1) is set
```

```
                The Information Element is used as Flow Key.

            Both bit scope(0) and flowKey(1) MUST NOT be set at the same
            time.  This combination is not allowed."
        REFERENCE
            "RFC 5101, Specification of the IP Flow Information
            Export (IPFIX) Protocol for the Exchange of IP Traffic Flow
            Information, Sections 2 and 3.4.2.1.

            RFC 5102, Information Model for IP Flow Information Export."
        ::= { ipfixTemplateDefinitionEntry 5 }

    --------------------------------------------------------------------
    -- 1.1.4: Export Table
    --------------------------------------------------------------------
    ipfixExportTable  OBJECT-TYPE
        SYNTAX      SEQUENCE OF IpfixExportEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "This table lists all exports of an IPFIX device.

            On Exporters, this table contains all exports grouped by
            Transport Session, Observation Domain Id, Template Id, and
            Metering Process represented by the
            ipfixMeteringProcessCacheId.  Thanks to the ipfixExportIndex,
            the exports can group one or more Transport Sessions to
            achieve a special functionality like failover management,
            load-balancing, etc.  The entries with the same
            ipfixExportIndex, ipfixObservationDomainId,
            and ipfixMeteringProcessCacheId define a Transport
            Session group.  If the Exporter does not use Transport
            Session grouping, then each ipfixExportIndex contains a
            single ipfixMeteringProcessCacheId and thus a singe
            Transport Session, and this session MUST have the member
            type primary(1).  Transport Sessions referenced in this
            table MUST have the ipfixTransportSessionDeviceMode
            exporting(1).

            On Collectors, this table is not needed."
        ::= { ipfixMainObjects 4 }

    ipfixExportEntry OBJECT-TYPE
        SYNTAX      IpfixExportEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "Defines an entry in the ipfixExportTable."
```

```
       INDEX       {
           ipfixExportIndex,
           ipfixMeteringProcessCacheId,
           ipfixTransportSessionIndex
       }
       ::= { ipfixExportTable 1 }

   IpfixExportEntry ::=
       SEQUENCE {
          ipfixExportIndex      Unsigned32,
          ipfixExportMemberType INTEGER
       }

   ipfixExportIndex OBJECT-TYPE
       SYNTAX      Unsigned32 (1..4294967295)
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "Locally arbitrary, but unique identifier of an entry in
           the ipfixExportTable.  The value is expected
           to remain constant from a re-initialization of the entity's
           network management agent to the next re-initialization.

           A common ipfixExportIndex between two entries from this
           table expresses that there is a relationship between the
           Transport Sessions in ipfixTransportSessionIndex.  The type
           of relationship is expressed by the value of
           ipfixExportMemberType."
       ::= { ipfixExportEntry 1 }

   ipfixExportMemberType OBJECT-TYPE
       SYNTAX      INTEGER {
                       unknown(0),
                       primary(1),
                       secondary(2),
                       parallel(3),
                       loadBalancing(4)
                   }
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "The type of a member Transport Session in a Transport
           Session group (identified by the value of ipfixExportIndex,
           ipfixObservationDomainId, and ipfixMeteringProcessCacheId).
           The following values are valid:

           unknown(0)
               This value MUST be used if the status of the group
```

```
                membership cannot be detected by the equipment.  This
                value should be avoided as far as possible.

            primary(1)
                This value is used for a group member that is used as
                the primary target of an Exporter.  Other group members
                (with the same ipfixExportIndex and
                ipfixMeteringProcessCacheId) MUST NOT have the value
                primary(1) but MUST have the value secondary(2).
                This value MUST also be specified if the Exporter does
                not support Transport Session grouping.  In this case,
                the group contains only one Transport Session.

            secondary(2)
                This value is used for a group member that is used as a
                secondary target of an Exporter.  The Exporter will use
                one of the targets specified as secondary(2) within the
                same Transport Session group when the primary target is
                not reachable.

            parallel(3)
                This value is used for a group member that is used for
                duplicate exporting, i.e., all group members identified
                by the ipfixExportIndex are exporting the same Records
                in parallel.  This implies that all group members MUST
                have the same membertype parallel(3).

            loadBalancing(4)
                This value is used for a group member that is used
                as one target for load-balancing.  This means that a
                Record is sent to one of the group members in this
                group identified by ipfixExportIndex.
                This implies that all group members MUST have the same
                membertype loadBalancing(4)."
        ::= { ipfixExportEntry 2 }

   --------------------------------------------------------------------
   -- 1.1.5: Metering Process Table
   --------------------------------------------------------------------
   ipfixMeteringProcessTable  OBJECT-TYPE
       SYNTAX      SEQUENCE OF IpfixMeteringProcessEntry
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "This table lists so-called caches used at the Metering
           Process to store the metering data of Flows observed at
           the Observation Points given in the
           ipfixObservationPointGroupReference.  The table lists the
```

```
           timeouts that specify when the cached metering data is
           expired.

           On Collectors, the table is not needed."
       ::= { ipfixMainObjects 5 }

   ipfixMeteringProcessEntry OBJECT-TYPE
       SYNTAX      IpfixMeteringProcessEntry
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "Defines an entry in the ipfixMeteringProcessTable."
       INDEX       { ipfixMeteringProcessCacheId }
       ::= { ipfixMeteringProcessTable 1 }

   IpfixMeteringProcessEntry ::=
       SEQUENCE {
           ipfixMeteringProcessCacheId              Unsigned32,
           ipfixMeteringProcessObservationPointGroupRef Unsigned32,
           ipfixMeteringProcessCacheActiveTimeout   Unsigned32,
           ipfixMeteringProcessCacheInactiveTimeout Unsigned32
       }

   ipfixMeteringProcessCacheId OBJECT-TYPE
       SYNTAX      Unsigned32 (1..4294967295)
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "Locally arbitrary, but unique identifier of an entry in the
           ipfixMeterinProcessTable.  The value is expected to remain
           constant from a re-initialization of the entity's network
           management agent to the next re-initialization."
       ::= { ipfixMeteringProcessEntry 1 }

   ipfixMeteringProcessObservationPointGroupRef OBJECT-TYPE
       SYNTAX      Unsigned32
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "The Observation Point Group Id that links this table entry
           to the ipfixObservationPointTable.  The matching
           ipfixObservationPointGroupId in that table gives the
           Observation Points used in that cache.  If the Observation
           Points are unknown, the
           ipfixMeteringProcessObservationPointGroupRef MUST be zero."
       ::= { ipfixMeteringProcessEntry 2 }

   ipfixMeteringProcessCacheActiveTimeout OBJECT-TYPE
```

```
       SYNTAX      Unsigned32
       UNITS       "seconds"
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "On the Exporter, this object contains the time after which a
           Flow is expired (and a Data Record for the template is sent)
           even though packets matching this Flow are still received by
           the Metering Process.  If this value is 0, the Flow is not
           prematurely expired."
       REFERENCE
           "RFC 5470, Architecture for IP Flow Information Export,
           Section 5.1.1, item 3."
       ::= { ipfixMeteringProcessEntry 3 }

   ipfixMeteringProcessCacheInactiveTimeout OBJECT-TYPE
       SYNTAX      Unsigned32
       UNITS       "seconds"
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "On the Exporter. this object contains the time after which a
           Flow is expired (and a Data Record for the template is sent)
           when no packets matching this Flow are received by the
           Metering Process for the given number of seconds.  If this
           value is zero, the Flow is expired immediately, i.e., a Data
           Record is sent for every packet received by the Metering
           Process."
       REFERENCE
           "RFC 5470, Architecture for IP Flow Information Export,
           Section 5.1.1, item 1"
       ::= { ipfixMeteringProcessEntry 4 }

   --------------------------------------------------------------------
   -- 1.1.6: Observation Point Table
   --------------------------------------------------------------------
   ipfixObservationPointTable  OBJECT-TYPE
       SYNTAX      SEQUENCE OF IpfixObservationPointEntry
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "This table lists the Observation Points used within an
           Exporter by the Metering Process. The index
           ipfixObservationPointGroupId groups Observation Points
           and is referenced in the Metering Process table.

           On Collectors this table is not needed."
       ::= { ipfixMainObjects 6 }
```

```
   ipfixObservationPointEntry OBJECT-TYPE
       SYNTAX      IpfixObservationPointEntry
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "Defines an entry in the ipfixObservationPointTable."
       INDEX       {
           ipfixObservationPointGroupId,
           ipfixObservationPointIndex
       }
       ::= { ipfixObservationPointTable 1 }

   IpfixObservationPointEntry ::=
       SEQUENCE {
           ipfixObservationPointGroupId           Unsigned32,
           ipfixObservationPointIndex             Unsigned32,
           ipfixObservationPointObservationDomainId Unsigned32,
           ipfixObservationPointPhysicalEntity    PhysicalIndexOrZero,
           ipfixObservationPointPhysicalInterface InterfaceIndexOrZero,
           ipfixObservationPointPhysicalEntityDirection INTEGER
       }

   ipfixObservationPointGroupId OBJECT-TYPE
       SYNTAX      Unsigned32 (1..4294967295)
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "Locally arbitrary, but unique identifier of an entry in the
           ipfixObservationPointTable.  The value is expected to remain
           constant from a re-initialization of the entity's network
           management agent to the next re-initialization.

           This index represents a group of Observation Points.

           The special value of 0 MUST NOT be used within this table
           but is reserved for the usage in the
           ipfixMeteringProcessTable.  An index of 0 for the
           ipfixObservationPointGroupReference index in that table
           indicates that an Observation Point is unknown or
           unspecified for a Metering Process cache."
       ::= { ipfixObservationPointEntry 1 }

   ipfixObservationPointIndex OBJECT-TYPE
       SYNTAX      Unsigned32 (1..4294967295)
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "Locally arbitrary, but unique identifier of an entry in the
```

```
          ipfixObservationPointTable.  The value is expected to remain
          constant from a re-initialization of the entity's network
          management agent to the next re-initialization.

          This index represents a single Observation Point in an
          Observation Point group."
      ::= { ipfixObservationPointEntry 2 }

   ipfixObservationPointObservationDomainId OBJECT-TYPE
      SYNTAX      Unsigned32
      MAX-ACCESS  read-only
      STATUS      current

      DESCRIPTION
          "The Id of the Observation Domain in which this
          Observation Point is included.

          The special value of 0 indicates that the Observation
          Points within this group cannot be applied to a single
          Observation Domain."
      REFERENCE
          "RFC 5101, Specification of the IP Flow Information Export
          (IPFIX) Protocol for the Exchange of IP
          Traffic Flow Information, Section 3.1."
      ::= { ipfixObservationPointEntry 3 }

   ipfixObservationPointPhysicalEntity OBJECT-TYPE
      SYNTAX      PhysicalIndexOrZero
      MAX-ACCESS  read-only
      STATUS      current
      DESCRIPTION
          "This object contains the index of a physical entity in
          the ENTITY MIB.  This physical entity is the given
          Observation Point.  If such a physical entity cannot be
          specified or is not known, then the object is zero."
      ::= { ipfixObservationPointEntry 4 }

   ipfixObservationPointPhysicalInterface OBJECT-TYPE
      SYNTAX      InterfaceIndexOrZero
      MAX-ACCESS  read-only
      STATUS      current
      DESCRIPTION
          "This object contains the index of a physical interface in
          the IF MIB.  This physical interface is the given
          Observation Point.  If such a physical interface cannot be
          specified or is not known, then the object is zero.

          This object MAY be used stand alone or in addition to
```

```
          ipfixObservationPointPhysicalEntity.  If
          ipfixObservationPointPhysicalEntity is not zero, this object
          MUST point to the same physical interface that is
          referenced in ipfixObservationPointPhysicalEntity.
          Otherwise, it may reference any interface in the IF MIB."
      ::= { ipfixObservationPointEntry 5 }

   ipfixObservationPointPhysicalEntityDirection OBJECT-TYPE
      SYNTAX      INTEGER {
                      unknown(0),
                      ingress(1),
                      egress(2),
                      both(3)
                  }
      MAX-ACCESS  read-only
      STATUS      current
      DESCRIPTION
          "The direction of the Flow that is monitored on the given
          physical entity.  The following values are valid:

          unknown(0)
              This value MUST be used if a direction is not
              known for the given physical entity.

          ingress(1)
              This value is used for monitoring incoming Flows on the
              given physical entity.

          egress(2)
              This value is used for monitoring outgoing Flows on the
              given physical entity.

          both(3)
              This value is used for monitoring incoming and outgoing
              Flows on the given physical entity."
      ::= { ipfixObservationPointEntry 6 }

   --------------------------------------------------------------------
   -- 1.1.7: Selection Process Table
   --------------------------------------------------------------------
   ipfixSelectionProcessTable OBJECT-TYPE
      SYNTAX      SEQUENCE OF IpfixSelectionProcessEntry
      MAX-ACCESS  not-accessible
      STATUS      current
      DESCRIPTION
          "This table contains Selector Functions connected to a
          Metering Process by the index ipfixMeteringProcessCacheId.
          The Selector Functions are grouped into Selection Processes
```

```
           by the ipfixSelectionProcessIndex.  The Selector Functions
           are applied within the Selection Process to the packets
           observed for the given Metering Process cache in increasing
           order implied by the ipfixSelectionProcessSelectorIndex.
           This means Selector Functions with lower
           ipfixSelectionProcessSelectorIndex are applied first.  The
           remaining packets are accounted for in Flow Records.

           Since IPFIX does not define any Selector Function (except
           selecting every packet), this is a placeholder for future
           use and a guideline for implementing enterprise-specific
           Selector Function objects.

           The following object tree should visualize how the
           Selector Function objects should be implemented:

           ipfixSelectorFunctions
           |
           +- ipfixFuncSelectAll
           |  |
           |  +- ipfixFuncSelectAllAvail (is the function available?)
           |
           +- ipfixFuncF2
           |  |
           |  +- ipfixFuncF2Avail (is the function F2 available?)
           |  |
           |  +- ipfixFuncF2Parameters (a table with parameters)
           ...
           |
           +- ipfixFunFn...

           If a Selector Function takes parameters, the MIB should
           contain a table with an entry for each set of parameters
           used at the Exporter."
       ::= { ipfixMainObjects 7 }

   ipfixSelectionProcessEntry OBJECT-TYPE
       SYNTAX      IpfixSelectionProcessEntry
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "Defines an entry in the ipfixSelectionProcessTable."
       INDEX       {
           ipfixMeteringProcessCacheId,
           ipfixSelectionProcessIndex,
           ipfixSelectionProcessSelectorIndex
       }
       ::= { ipfixSelectionProcessTable 1 }
```

```
   IpfixSelectionProcessEntry ::= SEQUENCE {
           ipfixSelectionProcessIndex            Unsigned32,
           ipfixSelectionProcessSelectorIndex    Unsigned32,
           ipfixSelectionProcessSelectorFunction OBJECT IDENTIFIER
       }

   ipfixSelectionProcessIndex OBJECT-TYPE
       SYNTAX      Unsigned32 (1..4294967295)
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "Locally arbitrary, but unique identifier of an entry in the
           ipfixSelectionProcessTable.  The value is expected to remain
           constant from a re-initialization of the entity's network
           management agent to the next re-initialization."
       ::= { ipfixSelectionProcessEntry 1 }

   ipfixSelectionProcessSelectorIndex OBJECT-TYPE
       SYNTAX      Unsigned32 (1..4294967295)
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "Index specifying the order in which the referenced
           ipfixSelctionProcessSelectorFunctions are applied to the
           observed packet stream within the given Selection Process
           (identified by the ipfixSelectionProcessIndex).  The
           Selector Functions are applied in increasing order, i.e.,
           Selector Functions with lower index are applied first."
       ::= { ipfixSelectionProcessEntry 2 }

   ipfixSelectionProcessSelectorFunction OBJECT-TYPE
       SYNTAX      OBJECT IDENTIFIER
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "The pointer to the Selector Function used at position
           ipfixSelectionProcessSelectorIndex in the list of Selector
           Functions for the Metering Process cache specified by the
           index ipfixMeteringProcessCacheId and for the given
           Selection Process (identified by the
           ipfixSelectionProcessIndex).

           This usually points to an object in the IPFIX SELECTOR MIB.
           If the Selector Function does not take parameters, then it
           MUST point to the root of the function subtree.  If the
           function takes parameters, then it MUST point to an entry
           in the parameter table of the Selector Function."
       ::= { ipfixSelectionProcessEntry 3 }
```

```
   -------------------------------------------------------------------
   -- 1.2.1: Transport Session Statistics Table
   -------------------------------------------------------------------
   ipfixTransportSessionStatsTable  OBJECT-TYPE
       SYNTAX      SEQUENCE OF IpfixTransportSessionStatsEntry
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "This table lists Transport Sessions statistics between
           Exporting Processes and Collecting Processes."
       ::= { ipfixStatistics 1 }

   ipfixTransportSessionStatsEntry OBJECT-TYPE
       SYNTAX      IpfixTransportSessionStatsEntry
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "Defines an entry in the ipfixTransportSessionStatsTable."
       AUGMENTS    { ipfixTransportSessionEntry }
       ::= { ipfixTransportSessionStatsTable 1 }

   IpfixTransportSessionStatsEntry ::=
       SEQUENCE {
           ipfixTransportSessionRate              Gauge32,
           ipfixTransportSessionPackets           Counter64,
           ipfixTransportSessionBytes             Counter64,
           ipfixTransportSessionMessages          Counter64,
           ipfixTransportSessionDiscardedMessages Counter64,
           ipfixTransportSessionRecords           Counter64,
           ipfixTransportSessionTemplates         Counter64,
           ipfixTransportSessionOptionsTemplates  Counter64,
           ipfixTransportSessionDiscontinuityTime TimeStamp
       }

   ipfixTransportSessionRate OBJECT-TYPE
       SYNTAX      Gauge32
       UNITS       "bytes/second"
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "The number of bytes per second received by the
           Collector or transmitted by the Exporter.  A
           value of zero (0) means that no packets were sent or
           received, yet.  This object is updated every second."
       ::= { ipfixTransportSessionStatsEntry 1 }

   ipfixTransportSessionPackets OBJECT-TYPE
       SYNTAX      Counter64
```

```
        UNITS       "packets"
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The number of packets received by the Collector
            or transmitted by the Exporter.
            Discontinuities in the value of this counter can occur at
            re-initialization of the management system and at other
            times as indicated by the value of
            ipfixTransportSessionDiscontinuityTime."
        ::= { ipfixTransportSessionStatsEntry 2 }

    ipfixTransportSessionBytes OBJECT-TYPE
        SYNTAX      Counter64
        UNITS       "bytes"
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The number of bytes received by the Collector
            or transmitted by the Exporter.
            Discontinuities in the value of this counter can occur at
            re-initialization of the management system and at other
            times as indicated by the value of
            ipfixTransportSessionDiscontinuityTime."
        ::= { ipfixTransportSessionStatsEntry 3 }

    ipfixTransportSessionMessages OBJECT-TYPE
        SYNTAX      Counter64
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The number of IPFIX Messages received by the
            Collector or transmitted by the Exporter.
            Discontinuities in the value of this counter can occur at
            re-initialization of the management system and at other
            times as indicated by the value of
            ipfixTransportSessionDiscontinuityTime."
        ::= { ipfixTransportSessionStatsEntry 4 }

    ipfixTransportSessionDiscardedMessages OBJECT-TYPE
        SYNTAX      Counter64
        MAX-ACCESS  read-only
        STATUS      current

        DESCRIPTION
            "The number of received IPFIX Message that are malformed,
            cannot be decoded, are received in the wrong order, or are
            missing according to the sequence number.
```

```
           If used at the Exporter, the number of messages that could
           not be sent due to, e.g., internal buffer overflows, network
           congestion, or routing issues.
           Discontinuities in the value of this counter can occur at
           re-initialization of the management system and at other
           times as indicated by the value of
           ipfixTransportSessionDiscontinuityTime."
       ::= { ipfixTransportSessionStatsEntry 5 }

   ipfixTransportSessionRecords OBJECT-TYPE
       SYNTAX      Counter64
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "The number of Data Records received by the Collector or
           transmitted by the Exporter.
           Discontinuities in the value of this counter can occur at
           re-initialization of the management system and at other
           times as indicated by the value of
           ipfixTransportSessionDiscontinuityTime."
       ::= { ipfixTransportSessionStatsEntry 6 }

   ipfixTransportSessionTemplates OBJECT-TYPE
       SYNTAX      Counter64
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "The number of Templates received or transmitted.
           Discontinuities in the value of this counter can occur at
           re-initialization of the management system and at other
           times as indicated by the value of
           ipfixTransportSessionDiscontinuityTime."
       ::= { ipfixTransportSessionStatsEntry 7 }

   ipfixTransportSessionOptionsTemplates OBJECT-TYPE
       SYNTAX      Counter64
       MAX-ACCESS  read-only
       STATUS      current

       DESCRIPTION
           "The number of Options Templates received or transmitted.
           Discontinuities in the value of this counter can occur at
           re-initialization of the management system and at other
           times as indicated by the value of
           ipfixTransportSessionDiscontinuityTime."
       ::= { ipfixTransportSessionStatsEntry 8 }

   ipfixTransportSessionDiscontinuityTime OBJECT-TYPE
```

```
       SYNTAX       TimeStamp
       MAX-ACCESS   read-only
       STATUS       current
       DESCRIPTION
           "The value of sysUpTime at the most recent occasion at which
           one or more of the Transport Session counters suffered a
           discontinuity.
           A value of zero indicates no such discontinuity has
           occurred since the last re-initialization of the local
           management subsystem."
       ::= { ipfixTransportSessionStatsEntry 9 }

   --------------------------------------------------------------------
   -- 1.2.2: Template Statistics Table
   --------------------------------------------------------------------
   ipfixTemplateStatsTable  OBJECT-TYPE
       SYNTAX      SEQUENCE OF IpfixTemplateStatsEntry
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "This table lists statistics objects per Template."
       ::= { ipfixStatistics 2 }

   ipfixTemplateStatsEntry OBJECT-TYPE
       SYNTAX      IpfixTemplateStatsEntry
       MAX-ACCESS  not-accessible
       STATUS      current
       DESCRIPTION
           "Defines an entry in the ipfixTemplateStatsTable."
       AUGMENTS    { ipfixTemplateEntry }
       ::= { ipfixTemplateStatsTable 1 }

   IpfixTemplateStatsEntry ::=
       SEQUENCE {
           ipfixTemplateDataRecords       Counter64,
           ipfixTemplateDiscontinuityTime TimeStamp
       }

   ipfixTemplateDataRecords OBJECT-TYPE
       SYNTAX      Counter64
       MAX-ACCESS  read-only
       STATUS      current
       DESCRIPTION
           "The number of Data Records that are transmitted or received
           per Template.
           Discontinuities in the value of this counter can occur at
           re-initialization of the management system, and at other
           times as indicated by the value of
```

```
            ipfixTemplateDiscontinuityTime."
        ::= { ipfixTemplateStatsEntry 1 }

    ipfixTemplateDiscontinuityTime OBJECT-TYPE
        SYNTAX       TimeStamp
        MAX-ACCESS   read-only
        STATUS       current
        DESCRIPTION
            "The value of sysUpTime at the most recent occasion at which
            the Template counter suffered a discontinuity.
            A value of zero indicates no such discontinuity has
            occurred since the last re-initialization of the local
            management subsystem."
        ::= { ipfixTemplateStatsEntry 2 }

    --------------------------------------------------------------------
    -- 1.2.3: Metering Process Statistics Table
    --------------------------------------------------------------------
    ipfixMeteringProcessStatsTable  OBJECT-TYPE
        SYNTAX      SEQUENCE OF IpfixMeteringProcessStatsEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "This table lists statistic objects that have data per
            Metering Process cache.

            On Collectors, this table is not needed."
        ::= { ipfixStatistics 3 }

    ipfixMeteringProcessStatsEntry OBJECT-TYPE
        SYNTAX      IpfixMeteringProcessStatsEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "Defines an entry in the ipfixMeteringProcessStatsTable."
        AUGMENTS    { ipfixMeteringProcessEntry }
        ::= { ipfixMeteringProcessStatsTable 1 }

    IpfixMeteringProcessStatsEntry ::=
        SEQUENCE {
            ipfixMeteringProcessCacheActiveFlows          Gauge32,
            ipfixMeteringProcessCacheUnusedCacheEntries   Gauge32,
            ipfixMeteringProcessCacheDataRecords          Counter64,
            ipfixMeteringProcessCacheDiscontinuityTime    TimeStamp
        }

    ipfixMeteringProcessCacheActiveFlows OBJECT-TYPE
        SYNTAX      Gauge32
```

```
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The number of Flows currently active at this cache."
        ::= { ipfixMeteringProcessStatsEntry 1 }

    ipfixMeteringProcessCacheUnusedCacheEntries   OBJECT-TYPE
        SYNTAX      Gauge32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The number of unused cache entries."
        ::= { ipfixMeteringProcessStatsEntry 2 }

    ipfixMeteringProcessCacheDataRecords OBJECT-TYPE
        SYNTAX      Counter64
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The number of Data Records generated.
            Discontinuities in the value of this counter can occur at
            re-initialization of the management system and at other
            times as indicated by the value of
            ipfixTemplateDiscontinuityTime."
        ::= { ipfixMeteringProcessStatsEntry 3 }

    ipfixMeteringProcessCacheDiscontinuityTime OBJECT-TYPE
        SYNTAX       TimeStamp
        MAX-ACCESS   read-only
        STATUS       current
        DESCRIPTION
            "The value of sysUpTime at the most recent occasion at which
            the Metering Process counter suffered a discontinuity.
            A value of zero indicates no such discontinuity has
            occurred since the last re-initialization of the local
            management subsystem."
        ::= { ipfixMeteringProcessStatsEntry 4 }

    ------------------------------------------------------------------
    -- 1.2.4: Selection Process Statistics Table
    ------------------------------------------------------------------
    ipfixSelectionProcessStatsTable OBJECT-TYPE
        SYNTAX      SEQUENCE OF IpfixSelectionProcessStatsEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "This table contains statistics for the Selector Functions
            connected to Metering Process by the index
```

```
            ipfixMeteringProcessCacheId.

            The indexes MUST match an entry in the
            ipfixSelectionProcessTable."
        ::= { ipfixStatistics 4 }

   ipfixSelectionProcessStatsEntry OBJECT-TYPE
        SYNTAX      IpfixSelectionProcessStatsEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "Defines an entry in the ipfixSelectionProcessStatsTable."
        AUGMENTS    { ipfixSelectionProcessEntry }
        ::= { ipfixSelectionProcessStatsTable 1 }

   IpfixSelectionProcessStatsEntry ::= SEQUENCE {
            ipfixSelectionProcessStatsPacketsObserved   Counter64,
            ipfixSelectionProcessStatsPacketsDropped    Counter64,
            ipfixSelectionProcessStatsDiscontinuityTime TimeStamp
        }

   ipfixSelectionProcessStatsPacketsObserved OBJECT-TYPE
        SYNTAX      Counter64
        MAX-ACCESS  read-only
        STATUS      current

        DESCRIPTION
            "The number of packets observed at the entry point of the
            function.  The entry point may be the Observation Point or
            the exit point of another Selector Function.
            Discontinuities in the value of this counter can occur at
            re-initialization of the management system and at other
            times as indicated by the value of
            ipfixSelectionProcessStatsDiscontinuityTime."
        ::= { ipfixSelectionProcessStatsEntry 1 }

   ipfixSelectionProcessStatsPacketsDropped OBJECT-TYPE
        SYNTAX      Counter64
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The number of packets dropped while selecting packets.
            Discontinuities in the value of this counter can occur at
            re-initialization of the management system and at other
            times as indicated by the value of
            ipfixSelectionProcessStatsDiscontinuityTime."
        ::= { ipfixSelectionProcessStatsEntry 2 }
```

```
   ipfixSelectionProcessStatsDiscontinuityTime OBJECT-TYPE
       SYNTAX       TimeStamp
       MAX-ACCESS   read-only
       STATUS       current
       DESCRIPTION
           "The value of sysUpTime at the most recent occasion at which
           one or more of the Selector counters suffered a
           discontinuity.
           A value of zero indicates no such discontinuity has
           occurred since the last re-initialization of the local
           management subsystem."
       ::= { ipfixSelectionProcessStatsEntry 3 }

   --==================================================================
   -- 2: Conformance Information
   --==================================================================
   ipfixCompliances OBJECT IDENTIFIER ::= { ipfixConformance 1 }
   ipfixGroups      OBJECT IDENTIFIER ::= { ipfixConformance 2 }

   --------------------------------------------------------------------
   -- 2.1: Compliance Statements
   --------------------------------------------------------------------
   ipfixCollectorCompliance MODULE-COMPLIANCE
       STATUS      current
       DESCRIPTION
           "An implementation that builds an IPFIX Collector
           that complies to this module MUST implement the objects
           defined in the mandatory group ipfixCommonGroup.

           The implementation of all objects in the other groups is
           optional and depends on the corresponding functionality
           implemented in the equipment.

           An implementation that is compliant to this MIB module
           is limited to use only the values TCP (6), UDP (17), and
           SCTP (132) in the ipfixTransportSessionProtocol object
           because these are the only protocol currently specified
           for usage within IPFIX (see RFC 5101)."
       MODULE  -- this module
       MANDATORY-GROUPS {
           ipfixCommonGroup
       }

       GROUP ipfixCommonStatsGroup
       DESCRIPTION
           "These objects should be implemented if the statistics
           function is implemented in the equipment."
       ::= { ipfixCompliances 1 }
```

```
   ipfixExporterCompliance MODULE-COMPLIANCE
       STATUS  current
       DESCRIPTION
           "An implementation that builds an IPFIX Exporter that
           complies to this module MUST implement the objects defined
           in the mandatory group ipfixCommonGroup.  The implementation
           of all other objects depends on the implementation of the
           corresponding functionality in the equipment."
       MODULE  -- this module

       MANDATORY-GROUPS {
               ipfixCommonGroup,
               ipfixExporterGroup
       }

       GROUP ipfixCommonStatsGroup
       DESCRIPTION
           "These objects should be implemented if the statistics
           function is implemented in the equipment."

       GROUP ipfixExporterStatsGroup
       DESCRIPTION
           "These objects MUST be implemented if statistical functions
           are implemented on the equipment."
       ::= { ipfixCompliances 2 }

   --------------------------------------------------------------------
   -- 2.2: MIB Grouping
   --------------------------------------------------------------------
   ipfixCommonGroup OBJECT-GROUP
       OBJECTS {
           ipfixTransportSessionProtocol,
           ipfixTransportSessionSourceAddressType,
           ipfixTransportSessionSourceAddress,
           ipfixTransportSessionDestinationAddressType,
           ipfixTransportSessionDestinationAddress,
           ipfixTransportSessionSourcePort,
           ipfixTransportSessionDestinationPort,
           ipfixTransportSessionSctpAssocId,
           ipfixTransportSessionDeviceMode,
           ipfixTransportSessionTemplateRefreshTimeout,
           ipfixTransportSessionOptionsTemplateRefreshTimeout,
           ipfixTransportSessionTemplateRefreshPacket,
           ipfixTransportSessionOptionsTemplateRefreshPacket,
           ipfixTransportSessionIpfixVersion,
           ipfixTransportSessionStatus,

           ipfixTemplateSetId,
```

```
            ipfixTemplateAccessTime,

            ipfixTemplateDefinitionIeId,
            ipfixTemplateDefinitionIeLength,
            ipfixTemplateDefinitionEnterpriseNumber,
            ipfixTemplateDefinitionFlags
        }
        STATUS      current

        DESCRIPTION
            "The main IPFIX objects."
        ::= { ipfixGroups 1 }

    ipfixCommonStatsGroup OBJECT-GROUP
        OBJECTS {
            ipfixTransportSessionRate,
            ipfixTransportSessionPackets,
            ipfixTransportSessionBytes,
            ipfixTransportSessionMessages,
            ipfixTransportSessionDiscardedMessages,
            ipfixTransportSessionRecords,
            ipfixTransportSessionTemplates,
            ipfixTransportSessionOptionsTemplates,
            ipfixTransportSessionDiscontinuityTime,

            ipfixTemplateDataRecords,
            ipfixTemplateDiscontinuityTime
        }
        STATUS      current
        DESCRIPTION
            "Common statistical objects."
        ::= { ipfixGroups 2 }

    ipfixExporterGroup OBJECT-GROUP
        OBJECTS {
            ipfixExportMemberType,

            ipfixMeteringProcessObservationPointGroupRef,
            ipfixMeteringProcessCacheActiveTimeout,
            ipfixMeteringProcessCacheInactiveTimeout,

            ipfixObservationPointObservationDomainId,
            ipfixObservationPointPhysicalEntity,
            ipfixObservationPointPhysicalInterface,
            ipfixObservationPointPhysicalEntityDirection,

            ipfixSelectionProcessSelectorFunction
        }
```

```
        STATUS      current
        DESCRIPTION
            "The main objects for Exporters."
        ::= { ipfixGroups 3 }

    ipfixExporterStatsGroup OBJECT-GROUP
        OBJECTS {
            ipfixMeteringProcessCacheActiveFlows,
            ipfixMeteringProcessCacheUnusedCacheEntries,
            ipfixMeteringProcessCacheDataRecords,
            ipfixMeteringProcessCacheDiscontinuityTime,

            ipfixSelectionProcessStatsPacketsObserved,
            ipfixSelectionProcessStatsPacketsDropped,
            ipfixSelectionProcessStatsDiscontinuityTime
        }
        STATUS      current
        DESCRIPTION
            "The statistical objects for Exporters."
        ::= { ipfixGroups 4 }

    END
```

8.2. IPFIX SELECTOR MIB Definition

```
    IPFIX-SELECTOR-MIB DEFINITIONS ::= BEGIN

    IMPORTS
        MODULE-IDENTITY, OBJECT-TYPE, mib-2
            FROM SNMPv2-SMI                               -- RFC2578
        TruthValue
            FROM SNMPv2-TC                                -- RFC2579
        MODULE-COMPLIANCE, OBJECT-GROUP
            FROM SNMPv2-CONF;                             -- RFC2580

    ipfixSelectorMIB MODULE-IDENTITY
        LAST-UPDATED "201110200000Z"         -- 20 October 2011
        ORGANIZATION "IETF IPFIX Working Group"
        CONTACT-INFO
            "WG charter:
              http://www.ietf.org/html.charters/ipfix-charter.html

            Mailing Lists:
              General Discussion: ipfix@ietf.org
              To Subscribe: http://www1.ietf.org/mailman/listinfo/ipfix
              Archive:
          http://www1.ietf.org/mail-archive/web/ipfix/current/index.html
```

```
           Editor:
             Thomas Dietz
             NEC Europe Ltd.
             NEC Laboratories Europe
             Network Research Division
             Kurfuersten-Anlage 36
             69115 Heidelberg
             Germany
             Phone: +49 6221 4342-128
             Email: Thomas.Dietz@nw.neclab.eu

             Atsushi Kobayashi
             NTT Information Sharing Platform Laboratories
             3-9-11 Midori-cho
             Musashino-shi
             180-8585
             Japan
             Phone: +81-422-59-3978
             Email: akoba@nttv6.net

             Benoit Claise
             Cisco Systems, Inc.
             De Kleetlaan 6a b1
             Degem 1831
             Belgium
             Phone:  +32 2 704 5622
             Email: bclaise@cisco.com

             Gerhard Muenz
             Technische Universitaet Muenchen
             Department of Informatics
             Chair for Network Architectures and Services (I8)
             Boltzmannstr. 3
             85748 Garching
             Germany
             Phone: +49 89 289-18008
             Email: muenz@net.in.tum.de
             URI:   http://www.net.in.tum.de/~muenz"
       DESCRIPTION
           "The IPFIX SELECTOR MIB module defined in this section
           provides the standard Filtering and Sampling functions that
           can be referenced in the ipfixSelectionProcessTable.  All
           standard Filtering and Sampling functions MUST be registered
           in the subtree under object ipfixSelectorFunctions
           (1.3.6.1.2.1.194.1.1).  The toplevel OIDs in the subtree
           under object ipfixSelectorFunctions MUST be registered in a
           subregistry maintained by IANA at
           http://www.iana.org/assignments/smi-numbers.
```

```
          New selector functions MUST be registered at IANA and are
          subject to Expert Review  RFC 5226, i.e., review by one of a
          group of experts designated by an IETF Area Director.  The
          group of experts MUST check the requested MIB objects for
          completeness and accuracy of the description.  Requests for
          MIB objects that duplicate the functionality of existing
          objects SHOULD be declined.  The smallest available OID
          SHOULD be assigned to a new MIB objects.  The specification
          of new MIB objects SHOULD follow the structure specified in
          RFC [NewRFCNumber] and MUST be published using a well-
          established and persistent publication medium.  The experts
          will initially be drawn from the Working Group Chairs and
          document editors of the IPFIX and PSAMP Working Groups.

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

          Redistribution and use in source and binary forms, with or
          without modification, is permitted pursuant to, and subject
          to the license terms contained in, the Simplified BSD
          License set forth in Section 4.c of the IETF Trust's
          Legal Provisions Relating to IETF Documents
          (http://trustee.ietf.org/license-info)."

   --  Note for RFC Editor: substitute [NewRFCNumber] with the newly
   --  assigned number.

   --  Revision history

       REVISION     "201110200000Z"         -- 20 October 2011
       DESCRIPTION
           "Update to MIB description to reflect updated registration
           of new Sampling and Filtering Functions."

       REVISION     "201003150000Z"         -- 15 March 2010
       DESCRIPTION
           "Initial version, published as RFC 5815."

       ::= { mib-2 194 }

   --*******************************************************************
   -- Top Level Structure of the MIB
   --*******************************************************************

   ipfixSelectorObjects     OBJECT IDENTIFIER
       ::= { ipfixSelectorMIB 1 }
   ipfixSelectorConformance OBJECT IDENTIFIER
       ::= { ipfixSelectorMIB 2 }
```

```
--==================================================================
-- 1: Objects used by all IPFIX implementations
--==================================================================
--------------------------------------------------------------------
-- 1.1: Packet Selector Functions for IPFIX
--------------------------------------------------------------------
ipfixSelectorFunctions OBJECT IDENTIFIER
    ::= { ipfixSelectorObjects 1 }

--------------------------------------------------------------------
-- 1.1.1: Function 1: Selecting All Packets
--------------------------------------------------------------------
ipfixFuncSelectAll OBJECT IDENTIFIER
    ::= { ipfixSelectorFunctions 1 }

ipfixFuncSelectAllAvail OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This object indicates the availability of the trivial
        function of selecting all packets.  This function is always
        available."
    ::= { ipfixFuncSelectAll 1 }

--==================================================================
-- 2: Conformance Information
--==================================================================
ipfixSelectorCompliances OBJECT IDENTIFIER
    ::= { ipfixSelectorConformance 1 }
ipfixSelectorGroups      OBJECT IDENTIFIER
    ::= { ipfixSelectorConformance 2 }

--------------------------------------------------------------------
-- 2.1: Compliance Statements
--------------------------------------------------------------------
ipfixSelectorBasicCompliance MODULE-COMPLIANCE
    STATUS  current
    DESCRIPTION
        "An implementation that builds an IPFIX Exporter that
        complies to this module MUST implement the objects defined
        in the mandatory group ipfixBasicGroup.  The implementation
        of all other objects depends on the implementation of the
        corresponding functionality in the equipment."
    MODULE  -- this module
    MANDATORY-GROUPS {
            ipfixSelectorBasicGroup
    }
```

```
       ::= { ipfixSelectorCompliances 1 }

   --------------------------------------------------------------------
   -- 2.2: MIB Grouping
   --------------------------------------------------------------------
   ipfixSelectorBasicGroup OBJECT-GROUP
       OBJECTS {
           ipfixFuncSelectAllAvail
       }

       STATUS      current
       DESCRIPTION
           "The main IPFIX objects."
       ::= { ipfixSelectorGroups 1 }

   END
```

9. Security Considerations

There are no management objects defined in this MIB module that have a MAX-ACCESS clause of read-write and/or read-create. So, if these MIB modules are implemented correctly, then there is no risk that an intruder can alter or create any management objects of these MIB modules via direct SNMP SET operations.

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

- ipfixTransportSessionTable - contains configuration data that might be sensitive because objects in this table may reveal information about the network infrastructure

- ipfixExportTable - contains configuration data that might be sensitive because object in this table may reveal information about the network infrastructure as well

- ipfixMeteringProcessTable - contains configuration data that might be sensitive because objects in this table may reveal information about the IPFIX Device itself

- ipfixObservationPointTable - contains configuration data that might be sensitive because objects in this table may reveal information about the IPFIX Device itself and the network infrastructure

- ipfixSelectorFunctions - currently contains no sensitive data but might want to be secured anyway since it may contain sensitive data in a future version

All other objects and tables contain no data that is considered sensitive.

SNMP versions prior to SNMPv3 did not include adequate security. Even if the network itself is secure (for example by using IPsec), even then, there is no control as to who on the secure network is allowed to access and GET/SET (read/change/create/delete) the objects in these MIB modules.

It is RECOMMENDED that implementers consider the security features as provided by the SNMPv3 framework (see [RFC3410] Section 8), including full support for the SNMPv3 cryptographic mechanisms (for authentication and privacy).

Further, deployment of SNMP versions prior to SNMPv3 is NOT RECOMMENDED. Instead, it is RECOMMENDED to deploy SNMPv3 and to enable cryptographic security. It is then a customer/operator responsibility to ensure that the SNMP entity giving access to an instance of these MIB modules is properly configured to give access to the objects only to those principals (users) that have legitimate rights to indeed GET or SET (change/create/delete) them.

10. IANA Considerations

The MIB module in this document uses the following IANA-assigned OBJECT IDENTIFIER values recorded in the SMI Numbers registry:

Descriptor	OBJECT IDENTIFIER value
ipfixMIB	{ mib-2 193 }
ipfixSelectorMIB	{ mib-2 194 }

NOTE TO RFC EDITOR: substitute ThisRFC with the RFC number of this document after assignment in the following section.

The IPFIX SELECTOR MIB registry as defined in [RFC5815] Section 10 will be removed by IANA as its use is discontinued with this document.

Further on, IANA will maintain a subregistry at http://www.iana.org/assignments/smi-numbers in which the toplevel OIDs in the subtree under object ipfixSelectorFunctions MUST be registered. The initial version of this subregistry should contain the following content:

Sub-registry Name: IPFIX-SELECTOR-MIB Functions
Reference: [ThisRFC]
Registration Procedures: Expert Review [RFC5226]

Prefix:
mib-2.ipfixSelectorMIB.ipfixSelectorObjects.ipfixSelectorFunctions
(1.3.6.1.2.1.194.1.1)

Decimal	Name	Description	Reference
1	ipfixFuncSelectAll	Select everything	[ThisRFC]

Additions to this subregistry are subject to Expert Review [RFC5226], i.e., review by one of a group of experts designated by an IETF Area Director. The group of experts MUST check the requested MIB objects for completeness and accuracy of the description. Requests for MIB objects that duplicate the functionality of existing objects SHOULD be declined. The smallest available OID SHOULD be assigned to new MIB objects. The specification of new MIB objects SHOULD follow the structure specified in Section 6 and MUST be published using a well-established and persistent publication medium. The experts will initially be drawn from the Working Group Chairs and document editors of the IPFIX and PSAMP Working Groups.

11. Acknowledgments

This document is a product of the IPFIX Working Group. The authors would like to thank the following persons: Paul Aitken for his detailed review, Dan Romascanu and the MIB doctors, and many more, for the technical reviews and feedback.

12. References

12.1. Normative References

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

[RFC2578] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.

[RFC2579] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Textual Conventions for SMIv2", STD 58, RFC 2579, April 1999.

[RFC2580] McCloghrie, K., Perkins, D., and J. Schoenwaelder, "Conformance Statements for SMIv2", STD 58, RFC 2580, April 1999.

[RFC4001] Daniele, M., Haberman, B., Routhier, S., and J. Schoenwaelder, "Textual Conventions for Internet Network Addresses", RFC 4001, February 2005.

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

[RFC3873] Pastor, J. and M. Belinchon, "Stream Control Transmission Protocol (SCTP) Management Information Base (MIB)", RFC 3873, September 2004.

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

[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.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

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

12.2. Informative References

[RFC3410] Case, J., Mundy, R., Partain, D., and B. Stewart, "Introduction and Applicability Statements for Internet-Standard Management Framework", RFC 3410, December 2002.

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

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

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

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

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

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

Authors' Addresses

Thomas Dietz (editor)
NEC Europe, Ltd.
NEC Laboratories Europe
Network Research Division
Kurfuersten-Anlage 36
Heidelberg 69115
DE

Phone: +49 6221 4342-128
Email: Thomas.Dietz@neclab.eu

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

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

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

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

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

Email: muenz@net.in.tum.de

IPFIX Working Group A. Johnson
Internet-Draft B. Claise
Intended status: Standards Track P. Aitken
Expires: May 3, 2012 Cisco System, Inc.
J. Schoenwaelder
Jacobs University Bremen
October 31, 2011

Exporting MIB Variables using the IPFIX Protocol
draft-johnson-ipfix-mib-variable-export-03

Abstract

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

This method requires an extension to the current IPFIX protocol. New Template Set and Options Template Sets are specified to allow the export of Simple Network Management Protocol (SNMP) MIB Objects along with IPFIX 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 May 3, 2012.

Copyright Notice

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

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

Table of Contents

1. Open Issues / To do list

- "timestamps, exporters, and other animals" -> see the mailing list.

- Question: index is an IPFIX IE that didn’t appear the flow record? Do we preclude this case?

- The value of the MIB OID acting as an index may not be of fixed length and may have no default length, for example the OID can be of type string or type MIB OID.

- "we can use the IE as an index if there is one and only one similar with that length in the Template Records". To be discussed.

- Add an MIB variable data type in the IANA considerations.

- use case: no index count and no index OID in the SNMP agent -> add this with the solution discussed with the DCM2.0 team.

- This also allows reduced size encoding for the indices.

- some TODO in the XML version:

  * write section: "Indexed MIB Objects with a mix of MIB OID and IPFIX Information Element"

  * insert example: "Using MIB Objects with IPFIX Structured Data"

- Describe how to choose between multiple instances of the required index field (eg, when the index is the egress interface for multicast). eg, rather than specifying the index IE by ID, we could specify it by number: the n’th field in the record.

- IPFIX Structured Data: how should it work? Add example to "sectionStructuredData".

- How does the example in 5.5 work (ifOutQLen indexed by: ifIndex) since ifIndex is not present in the record?

- How does the example in 5.8.2 work, since the ifName is indexed by ifIndex which comes after - so the value is not already known.

- Improve the examples: Add an example with the mix of IPFIX IE and OID in sectionUseIndexedwithaMixofOIDAndIPFIXIE.

o RFC 5610: explain what needs to be updated.

o ID to name mappings? -> use this for an example in section 5.

o Fully specify the "MIBObjectIdentifierMarker" IE in the IANA section.

o What does this mean? : "(Consider the counter synchronisation issue, non-key info should be static)".

o Tidy up the XML.

o (JS) Do we need to add something about the contextEngineID and contextName? Optionally associate context with template via options Could be done with common properties or in a flow record However, do we limit all MIB variables in a Template Record to a single context? 3 cases:

  1. if a simple SNMP agent, no contextEngineID and contextName, because it’s the default

  2. the context information is valid for the entire flow record

  3. the context information is specific for each IE within the entire flow record

  question regarding 3.: only one context for an entire flow or can a flow record export MIB OID from different context? (JS): ask the IPFIX mailing list. (BC): ask internally in Cisco Action: complete the "Identifying the SNMP Context" section

o (JS) Inacio’s figure: send email to the mailing list.

2. Introduction

There is growing interest in using IPFIX as a push mechanism for exporting management information. Using a push protocol such as IPFIX instead of a polling protocol like SNMP is especially interesting in situations, where large chunks of repetitive data need to be exported periodically.

While initially targeted at different problems, there is a large parallel between the information transported via IPFIX and SNMP. Furthermore, certain Management Information Base (MIB) objects are highly relevant to flows as they are understood today. For example, in the IPFIX information model [RFC5102], Information Elements coming from the SNMP world have already been specified, e.g., ingressInterface and egressInterface both refer to the ifIndex defined in [RFC2863].

Rather than mapping existing MIB objects to IPFIX Information Elements on a case by case basis, it would be advantageous to enable the export of any existing or future MIB objects as part of an IPFIX Flow Record. This way, the duplication of data models [RFC3444], both as SMI MIB objects and IPFIX Information Elements, out of the same information model [RFC3444] would be avoided.

In this document, new Template Sets for Flow Records and Options Records are specified to allow Templates to contain any combination of fields defined by traditional IPFIX Information Element(s) and/or MIB Object Identifier(s). The MIB Object Identifiers can reference either non-indexed or indexed MIB object(s). Note that the enterprise-specific MIB Object Identifiers are also supported.

When an indexed MIB object is exported, a method to identify how that MIB object was indexed is specified so that the full meaning of the information being exported can be conveyed. The specifications encompasses the different index types for the MIB Objects Identifier: indexed by one or multiple MIB variable(s), indexed by one or multiple IPFIX IE(s), indexed by a mix of MIB variable(s) and IPFIX IE(s). A set of example use cases is used to illustrate how these specifications can be used.

3. Motivation and Architectural Model

Most Flow Records contain the ingressInterface and/or the egressInterface Information Element. These Information Elements carry an ifIndex value, a MIB object defined in [RFC2863]. In order to retrieve additional information about the identified interface, a Collector could simply poll relevant objects from the device running the Exporter via SNMP, however, that approach has several problems:

- o It requires implementing a mediation function between two data models, i.e., MIB objects and IPFIX Information Elements.

- o Confirming the validity of simple mappings (e.g., ifIndex to ifName) requires to either check on a regular basis that the Exporter’s network management system did not reload, or to impose ifIndex persistence across an Exporter’s reload.

- o Synchronization problems occur since counters carried in Flow Records and counters carried in SNMP messages are retrieved from the Exporter at different points in time and thus can’t be correlated. In the best case, assuming very tight integration of an IPFIX Collector with and SNMP polling engine, SNMP data is retrieved shortly after Data Records have been received, which implies the sum of the active or inactive timeouts (if not null) plus the time to export the Flow Record to the Collector. If, however, the SNMP data is retrieved by a generic Network Management Station (NMS) polling interface statistics, then the time lag between IPFIX counters and SNMP counters can be significant.

The intended scope of this work is the addition of MIB variable(s) to IPFIX Information Elements in Flow Records, in order to complement the Flow Records with useful and already standardized information. More specifically, the case of an existing Template Record, which needed to be augmented with some MIB variables whose index was already present in the Template Record as IPFIX IE: typically, a 7-tuple Flow Record containing the ingressInterface IE, augmented by interface counters [RFC2863], which are indexed by the respective ingressInterface values in the Flow Records.

The intended goal of this work is not a replacement of SNMP notifications, even if the specifications in this document could potentially allow this. Since IPFIX is a push mechanism, initiated from the Exporter with no acknowledgment method, this specification does not provide the ability to execute configuration changes.

The Distributed Management Expression MIB [RFC2982], which is a mechanism to create new MIB variables based on the content of existing ones, could also be advantageous in this context of this specification. Indeed, newly created MIB object (for example, the link utilization MIB variable), created with the Distributed Management Expression MIB [RFC2982] could nicely complement Flow Records.

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

```
  +------+  +-------+  +.........+  +.....+
  | SNMP |  | IPFIX |  : NETCONF :  : CLI :
  +------+  +-------+  +.........+  +.....+
      |         |           |          |
+---------------------------------------------+
| Instrumentation (specified in MIB modules) |
+---------------------------------------------+
```

Figure 1: Architectural Model

The overall architectural model is depicted in Figure 1. The IPFIX Exporter accesses the device's instrumentation, which follows the specifications contained in MIB modules. Other management interfaces such as NETCONF or the device's Command Line Interface (CLI) may provide access to the same instrumentation.

4. Terminology

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

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

MIB Object Identifier (MIB OID)

> An ASCII character sequences of decimal non-negative sub-identifier values. Each sub-identifier value MUST NOT exceed 2^32-1 (4294967295) and MUST NOT have leading zeros. Sub-identifiers are separated by single dots and without any intermediate whitespace.

MIB Object Identifier Information Element

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

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 [RFC2119].

5. MIB OID Extended Template Formats

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

5.1. MIB OID Extended Template Record Format

The format of the MIB Object Identifier Extended Template Record is shown in Figure 2. It consists of a Template Record Header and one or more Field Specifiers.

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

Figure 2: MIB Object Identifier Extended Template Record Format

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

The format of the Template Record Header is shown in Figure 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Template ID (> 255)   |         Field Count           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: Template Record Header Format

Where:

Template ID

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

Field Count

Number of all fields in this Template Record.

At this level of detail the layout of the Template Record Format, as specified in [RFC5101], and the MIB Object Identifier Extended

Template Record Format are identical. It is only the structure of the Field Specifiers that is different (see Section 5.3).

5.2. MIB OID Extended Options Template Record Format

The format of the MIB Object Identifier Extended Options Template Record is shown in Figure 4. It consists of an Options Template Record Header and one or more Field Specifiers.

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

Figure 4: MIB Object Identifier Options Extended Template Record Format

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

The format of the Options Template Record Header is shown in Figure 5.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Template ID (> 255)   |         Field Count           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Scope Field Count        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: Options Template Record Header Format

Where:

Template ID

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

Field Count

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

Scope Field Count

Number of scope fields in this Options Template Record. The Scope Fields are normal Fields except that they are interpreted as Scope at the Collector. The Scope Field Count MUST NOT be zero for an Options Template Record.

As with the Template Record Format, the only difference between the standard Options Template Record Format as defined in [RFC5101] and the MIB Object Identifier Extended Template Options Record Format is the structure of the Field Specifiers (see Section 5.3).

Both indexed and non-indexed MIB Objects may be used as scope fields in an IPFIX Options Template Record. Each scope MIB object is included in the IPFIX Scope Field Count. When indexed MIB Objects are used, the index information is not included in the Scope Field Count since the size of the index information is already specified in the MIB Object’s "index count" field (see Section 5.3.3). Examples are given in Section 6.8.

5.3. MIB OID Extended Field Specifier Format

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

The Field Specifier formats are shown in Figure 6 to Figure 9 below.

5.3.1. Standard Field Specifier Format

The Field Specifier format in Figure 6, along with the associated definitions, has been copied from [RFC5101], for an easier comparison with the MIB Object Identifier Extended Field Specifier Format in Figure 7 through Figure 9.

When exporting an IANA-assigned and/or enterprise-specific IPFIX Information Element identifier, the Field Specifier Format is the same as shown below.

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |E|  Information Element ident. |        Field Length           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                      Enterprise Number                        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 6: Standard Field Specifier format

Where:

E

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

Information Element identifier

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

Field Length

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

Enterprise Number

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

5.3.2. Extended Field Specifier Format for a non-indexed MIB Object

When a MIB object is to be exported, a special Information Element value is used to show that the extended Field Specifier is being used, as shown in Figure 7:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|E|         MIB OID IE          |         Field Length          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Index Count = 0|  MIB OID Len  |    MIB Object Identifier ...  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              ... MIB Object Identifier continued              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: MIB Object Identifier Extended Field Specifier Format for a non-indexed MIB Object with an OID length < 255

Where:

E

Enterprise bit. This is the first bit of the Field Specifier. The value is always set to 0 for the MIB Object Identifier Extended Field Specifier Format, even if the MIB Object Identifier is enterprise-specific, because the MIB OID IE is an IANA standard field and is not enterprise-specific.

MIB OID IE

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

Field Length

The definition is as [RFC5101]. Note that the Field Length can be expressed using reduced size encoding per [RFC5101].

Index Count

The number of indices for a MIB object. Set to zero for a non-indexed MIB object.

MIB Object Identifier Length

The length of the textual representation of the MIB Object Identifier that follows. This is encoded in the same manner as the variable length encoding in [RFC5101]. If the length of the MIB Object Identifier is greater than or equal to 255 octets, the length is encoded into 3 octets before the MIB Object Name, where the first octet is 255 and the length is carried in the second and third octets as shown in Figure 8. If the MIB Object Identifier is longer than 254 characters then the length MUST be extended.

MIB Object Identifier

The textual representation of a MIB object identifier as defined in Section 4.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|E|         MIB OID IE          |        Field Length           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Index Count = 0|      255      | MIB Object Identifier Length  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  MIB Object Identifier ...                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              ... MIB Object Identifier continued              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8: MIB Object Identifier Extended Field Specifier Format for a non-indexed MIB Object with an OID length >= 255

5.3.3. Extended Field Specifier Format for an Indexed MIB Object, With an MIB OID as Index

The mechanism for "Extended Field Specifier Format for non-indexed MIB Object" in Section 5.3.2 can be used for exporting any MIB objects, including indexed MIB objects. However, per the nature of indexing in MIB module, every indexed object is specified by a new MIB Object Identifier, which in turn implies that a new Template Record must be used for every indexed object. For example, the ifInOctets for the interface represented by the interface ifIndex 1 is ifInOctets.1, the ifInOctets for the interface represented by the interface ifIndex 2 is ifInOctets.2, ... This makes the export mechanism for "Extended Field Specifier Format for non-indexed MIB Object" inefficient when used for indexed MIB objects. An example is shown in Section 6.1.

When an indexed MIB object is exported in IPFIX, the meaning of the exported value of each index MUST be identified. This index (or indices) MUST be a MIB Object Identifier (this section) or an IPFIX Information Element (see Section 5.3.4).

A MIB Object Identifier MAY be used as an index and sent as described in Figure 9. However, if a MIB Object Identifier with an index is used as an index then its indices will not be identified.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|E|         MIB OID IE          |         Field Length          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Index Count  |  MIB OID Len  |   MIB Object Identifier ...   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              ... MIB Object Identifier continued              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|E|         MIB OID IE          |      Index Field length       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Idx MIB OID Len|        Index MIB Object Identifier ...        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         ... Index MIB Object Identifier continued ...         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                               ...
```

Figure 9: MIB Object Identifier Extended Field Specifier Format with a MIB Index using a normal MIB Object Identifier as index

Where:

E

Enterprise bit. This is the first bit of the Field Specifier. The value is always set to 0 for the MIB Object Identifier Extended Field Specifier Format, even if the MIB Object Identifier is enterprise-specific, because the MIB OID IE is an IANA standard field and is not enterprise-specific.

MIB OID IE

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

Field Length

The definition is as [RFC5101]. Note that the Field Length can be expressed using reduced size encoding per [RFC5101].

Index Count

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

MIB Object Identifier Length

The length of the textual representation of the MIB Object Identifier that follows. This is encoded in the same manner as the variable length encoding in [RFC5101]. If the length of the MIB Object Identifier is greater than or equal to 255 octets, the length is encoded into 3 octets before the MIB Object Name Where the first octet is 255 and the length is carried in the second and third octets (as shown in Figure 8). If the MIB Object Identifier is longer than 254 characters then the length MUST be extended.

MIB Object Identifier

The textual representation of a MIB object identifier as defined in Section 4. For any indices identified using Information Elements the Enterprise bit can be 1, indicating that an Enterprise Number will follow the Information Element.

Index Field Length

The length of the encoded index field, in octets, per the Field Length definition in [RFC5101]. Note that the Index Field Length can be expressed using reduced size encoding per [RFC5101].

Index MIB Object Identifier Length

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

Index MIB Object Identifier

The textual representation of a MIB object identifier as defined in Section 4.

5.3.4. Extended Field Specifier Format for an Indexed MIB Object, With an IPFIX IE as Index

A possible optimization for the Extended Field Specifier Format for an Indexed MIB Object as specified in Section 5.3.3 is to use an existing IPFIX Information Element, which is already present in the Flow definition, as the index for indexed MIB Object. On the top not repeating the index, the primary advantage is to make a clear link between the Flow Record values and the MIB variable index.

For example, if a Flow Record definition contains the source IP address, the destination IP address, and the ingressInterface Information Element as Flow Keys, this implies that the IP address pairs are seen on that specific interface. If the ifInOctets, indexed by that specific interface, is added to the Flow Record, it’s clear from the Flow Record, that the ifInOctets is related to the same interface. If the ifInOctets was indexed by the ifIndex (as specified in Section 5.3.3), the Collector would have to hardcode that the semantic of ifIndex MIB variable is equivalent to the ingressInterface Information Element.

When an indexed MIB object is exported in IPFIX, the index (or indices) MAY be an IPFIX Information Element(s). Note that this/ these IPFIX Information Element(s) MAY be an enterprise-specific Information Element.

Indexed MIB Objects, with IPFIX Information Elements as index, are exported as shown in Figure 10.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|E|         MIB OID IE          |         Field Length          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Index Count  |  MIB OID Len  |    MIB Object Identifier ...  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              ... MIB Object Identifier continued              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|E| Index Information Element 1 |       Index 1 Length          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|E| Index Information Element 2 |       Index 2 Length          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Index Information Element 2 Enterprise Number           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                             ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|E| Index Information Element N |       Index N Length          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Index Information Element N Enterprise Number           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 10: MIB Object Identifier Extended Field Specifier Format with an indexed MIB Object using an IPFIX Information Element as Index

Where:

E

Enterprise bit. This is the first bit of the Field Specifier. The value is always set to 0 for the MIB Object Identifier Extended Field Specifier Format, even if the MIB Object Identifier is enterprise-specific, because the MIB OID IE is an IANA standard field and is not enterprise-specific.

MIB OID IE

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

Field Length

The definition is as [RFC5101]. The Field Length does not include the length of the index fields, since these are specified separately. Note that the Field Length can be expressed using reduced size encoding per [RFC5101].

Index Count

The number of indices for a MIB object, and zero for a non-indexed MIB object. The index count MUST be consistent with the INDEX definition of the corresponding MIB module.

MIB Object Identifier Length

The length of the textual representation of the MIB Object Identifier that follows. This is encoded in the same manner as the variable length encoding in [RFC5101]. If the length of the MIB Object Identifier is greater than or equal to 255 octets, the length is encoded into 3 octets before the MIB Object Name where the first octet is 255 and the length is carried in the second and third octets (as shown in Figure 8). If the MIB Object Identifier is longer than 254 characters then the length MUST be extended.

MIB Object Identifier

The textual representation of a MIB object identifier as defined in Section 4.

Index Information Element 1..N

The Information Element(s) that are used as indices for the MIB Object Identifier.

Regular IEs, enterprise-specific IEs and non-indexed MIB object identifiers may all be used as indices. However, indexed MIB object identifiers may not be used as indices because SNMP doesn’t support hierarchical indexing.

Index 1..N Length

The respective index lengths for the Information Element(s) 1..N

5.4. Indices Considerations

When using an Indexed MIB Object, the Template Record contains the index/indices length. In some cases, this index/indices information might be redundant in the export information. For example, when the index is an Information Element already contained in the Template Record, the length is already part of the Template Record, and available to the Collecting Process for decode, as shown in the example in Section 6.6. A second example in Section 6.8 is when a specific MIB OID is already part of the Template Record as a standalone MIB object in a Template Record, and also reused as an index.

However, there are two cases where the index length is required. Therefore, for consistent decoding on the Collecting Process, the Index Length is always specified next to the index.

Situation 1: When a non-indexed MIB object is used as an index, and doesn’t appear as a standalone MIB object in the Template Record, the Collecting Process might not want, per design, to access the MIB modules in order to find the length of the value for a particular MIB OID.

Situation 2: A Template Record might contain two similar Information Elements with different encoding lengths even if this situation is an unlikely real-world scenario), while an Indexed MIB Object might want to refer to one of this Information Element as the index. However, without clearly specifying the index length, the Collecting Process would not know which length to decode the index with.

When an Information Element is used as index, there MUST be one and only one similar Information Element with the exact same length in the Template Record, so that the Collecting Process knows which Information Element value from the Flow Records to match. Note that this rule also implies that the reduced size encoding [RFC5101] of the Information Element in the index compared to the Information Element in the Template Record is not allowed. If the Collecting Process can not determine clearly which Information Element value to chose as the index because there are two (or more) Information Elements with the same length, then index MUST specified as the MIB Object Identifier.

An indexed MIB object MAY be indexed by a mix of MIB OID(s) and IPFIX Information Element(s)

5.5. Identifying the SNMP Context

Each MIB OID is looked up in a specific context, usually the default context. If exporting a MIB OID value that isn’t in the default context then the context string MUST be identified and associated with the MIB OID. This can be done on a per template basis by exporting an Options Template Record.

A new IPFIX Information Element, "MIBObjectIdentifierMarker" has been allocated for this purpose. See Section 11.

5.6. Template Management

Templates are managed as per [RFC5101].

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

The Set ID field MUST contain the value TBD2 for any Option Template Set that contains a MIB Object Identifier Extended Field Specifier. The Template Withdrawal Message for such an Option Template must also use a Set ID field containing the value TBD2.

6. Example Use Cases

6.1. Without Using the Specifications in this Document

This example shows the need for indexed MIB objects using the example of exporting ifInOctets from Section 5.3.3.

A Template Record for exporting the ifInOctets for the interface represented by the interface ifIndex 1 (i.e., ifInOctets.1) is shown in Figure 11. While this may be useful for exporting the single ifInOctets.1 field, clearly additional Templates are required in order to export ifInOctets.2, ifInOctets.3, etc. Therefore Indexed MIB objects (per Section 5.3.3) are required in order to export arbitrary ifInOctets.x.

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |          Set ID = TBD1        |         Length = 36           |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |        Template ID = 256      |        Field Count = 1        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0|IE=MIBObjectIdentifierMarker |        Field Length = 4       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |Index Count = 0|MIB OID Len=22 |    MIB Object Identifier ...  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |     ... MIB Object Identifier = "1.3.6.1.2.1.2.2.1.10.1"      |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |           ...  MIB Object Identifier continued ...            |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |           ...  MIB Object Identifier continued ...            |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |           ...  MIB Object Identifier continued ...            |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |           ...  MIB Object Identifier continued                |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 11: Template for exporting ifInOctets.1

6.2. Non-indexed MIB Object: Established TCP Connections

The number of established TCP connections of a remote network device could be monitored by configuring it to periodically export the number of established TCP connections to a centralized Collector. In this example, the Exporter would export an IPFIX Message every 30 minutes that contained Data Records detailing the number of established TCP connections.

The table of data that is to be exported looks like:

TIMESTAMP	ESTABLISHED TCP CONN.
StartTime + 0 seconds	10
StartTime + 60 seconds	14
StartTime + 120 seconds	19
StartTime + 180 seconds	16
StartTime + 240 seconds	23
StartTime + 300 seconds	29

Table 1: Established TCP Connections

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

1. flowStartSeconds from [RFC5102], Information Element 150: The absolute timestamp of the first packet of this Flow.

2. tcpCurrEstab from [RFC4022], Object ID "1.3.6.1.2.1.6.9": The number of TCP connections for which the current state is either ESTABLISHED or CLOSE-WAIT.

Figure 12 shows the exported Template Set detailing the Template Record for exporting the number of established TCP connections (see Section 6.2).

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |          Set ID = TBD1        |          Length = 33          |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |        Template ID = 257      |        Field Count = 2        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0|   IE = flowStartSeconds     |        Field Length = 4       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0|IE=MIBObjectIdentifierMarker |        Field Length = 4       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |Index Count = 0|MIB OID Len=15 |    MIB Object Identifier ...  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |          ... MIB Object Identifier = "1.3.6.1.2.1.6.9"        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |            ...  MIB Object Identifier continued ...           |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |            ...  MIB Object Identifier continued ...           |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |            ... |
 +-+-+-+-+-+-+-+-+
```

Figure 12: Example of tcpCurrEstab Template Set

Figure 13 shows the start of the Data Set for exporting the number of established TCP connections (see Section 6.2).

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Set ID = 257         |         Length = 52           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    StartTime +   0 seconds                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              10                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    StartTime +  60 seconds                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              14                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    StartTime + 120 seconds                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              19                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    StartTime + 180 seconds                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              16                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    StartTime + 240 seconds                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              23                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    StartTime + 300 seconds                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              29                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 13: Example of tcpCurrEstab Data Set

6.3. Enterprise Specific MIB Object: Detailing CPU Load History

For the sake of demonstrating a enterprise-specific MIB object, a non-indexed MIB object is chosen for simplicity. The CPU Usage of a remote network device could be monitored by configuring it to periodically export CPU usage information, i.e. the cpmCPUTotal1minRev from the proprietary CISCO-PROCESS-MIB, Object ID "1.3.6.1.4.1.9.9.109.1.1.1.1.7", to a centralized Collector. In this example, the Exporter would export an IPFIX Message every 30 minutes that contained Data Records detailing the CPU 1 minute busy average at 1 minute intervals.

The table of data that is to be exported looks like:

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

Table 2: CPU Usage Data

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

1. flowStartSeconds from [RFC5102], Information Element 150: The absolute timestamp of the first packet of this Flow.

2. cpmCPUTotal1minRev, the overall CPU busy percentage in the last one-minute period

Figure 14 shows the exported Template Set detailing the Template Record for exporting CPU Load (see Section 6.3).

```
      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 = TBD1        |          Length = 47          |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |        Template ID = 258      |        Field Count = 2        |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |0|   IE = flowStartSeconds     |        Field Length = 4       |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |0|IE=MIBObjectIdentifierMarker |        Field Length = 1       |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |Index Count = 0|MIB OID Len=29 |    MIB Object Identifier ...  |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |  ... MIB Object Identifier = "1.3.6.1.4.1.9.9.109.1.1.1.1.7"  |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |           ...  MIB Object Identifier continued ...            |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |           ...  MIB Object Identifier continued ...            |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |           ...  MIB Object Identifier continued ...            |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |           ...  MIB Object Identifier continued ...            |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |           ...  MIB Object Identifier continued ...            |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |           ... MIB Object Identifier continued |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 14: Example of CPU Load Template Set

Note that although cpmCPUTotal1minRev is 32 bits long, reduced size encoding ([RFC5101]) has been used to encoded it within a single octet.

This example stresses that, even though the OID cpmCPUTotal1minRev is enterprise-specific, the E bit for the MIBObjectIdentifierMarker is set to "0" since the "MIBObjectIdentifierMarker" Information Element is not enterprise-specific.

The corresponding Data Set does not add any value for this example, and is therefore not displayed.

6.4. Indexed MIB Object with an OID: Output Interface Queue Size in PSAMP Packet Report

Following on the example from the previous section (see Section 6.6), if the Template Record for the example Data Record does not contain the egressInterface, the ifOutQLen must be indexed by the ifIndex interface index as detailed in the IF-MIB [RFC2863]:

The Template Record for the example Data Record contains the following Information Elements:

1. sourceIPv4Address

2. destinationIPv4Address

3. totalLengthIPv4

4. ifOutQLen indexed by: ifIndex

Figure 15 shows the exported Template Set detailing the Template for exporting a PSAMP Report with Interface Output Queue Length (ifOutQLen) but using the ifIndex MIB object as the exported index.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Set ID = TBD1        |         Length = 70           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Template ID = 259      |        Field Count = 4        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|   IE = sourceIPv4Address    |        Field Length = 4       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0| IE = destinationIPv4Address |        Field Length = 4       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|    IE = totalLengthIPv4     |        Field Length = 4       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|IE=MIBObjectIdentifierMarker |        Field Length = 1       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Index Count=1 |MIB OID Len=20 |    MIB Object Identifier ...  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      ... MIB Object Identifier = "1.3.6.1.2.1.2.2.1.21"       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           ...  MIB Object Identifier continued ...            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           ...  MIB Object Identifier continued ...            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           ...  MIB Object Identifier continued ...            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| ... MIB OID continued         |0|IE=MIBObjectIdentifierMarker |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  1.3.6.1.2.1.2.2.1.1 length   |MIB OID Len=19 | MIB Obj ID ...|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       MIB Object Identifier = "1.3.6.1.2.1.2.2.1.1" ...       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           ...  MIB Object Identifier continued ...            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           ...  MIB Object Identifier continued ...            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           ...  MIB Object Identifier continued ...            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| ... MIB Object Identifier cont|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 15: Example of a Template for a PSAMP Report with ifOutQLen using ifIndex from IF-MIB [RFC2863] as an index

Note that IPFIX reduced size encoding [RFC5101] has been used in this example to express ifOutQLen in a single octet, rather than the 32 bits specified in the IF-MIB [RFC2863].

The corresponding IPFIX Data Record is shown in Figure 16. For the sake of the example, the interface index of "Eth 1/0" is 15 and the interface index of "Eth 1/1" is 16.

```
 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 = 259         |         Length = 72           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          192.0.2.1                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          192.0.2.3                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             150                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      45       |                    15 ...                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      ...      |                 192.0.2.4 ...                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      ...      |                 192.0.2.9 ...                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      ...      |                    350 ...                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      ...      |       45      |            15 ...             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             ...               |        192.0.2.3 ...          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             ...               |        192.0.2.9 ...          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             ...               |             650 ...           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             ...               |       23      |    ...        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 ... 15                        |    ...        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 ... 192.0.2.4                 |    ...        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 ... 192.0.2.6                 |    ...        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 ...  350                      |       0       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              16                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 16: Example of PSAMP Packet Report with the ifOutQLen using ifIndex from IF-MIB [RFC2863] as an index

6.5. Indexed MIB Object with Two OIDs: The ipIfStatsInForwDatagrams

MIB objects may be indexed by multiple indices. Note that all the indices apply to the MIB object, i.e. index 2 is not an index of index 1.

This example shows the export of ipIfStatsInForwDatagrams from the IP-MIB [RFC4293] indexed by the ipIfStatsIPVersion and ipIfStatsIfIndex which are provided as scope fields in an IPFIX option. Note that since these fields are used as indices for ipIfStatsInForwDatagrams, they don't need their own indices to be identified.

The Options Template Record for the example Data Record contains the following Information Elements:

1. ipIfStatsIPVersion (1.3.6.1.2.1.4.31.3.1.1) (scope field)

2. ipIfStatsIfIndex (1.3.6.1.2.1.4.31.3.1.2) (scope field)

3. ipIfStatsInForwDatagrams (1.3.6.1.2.1.4.31.3.1.12) (non-scope field) indexed by ipIfStatsIPVersion and ipIfStatsIfIndex

Figure 17 shows the exported Options Template Set.

```
  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 = TBD2         |          Length = 146         |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |       Template ID = 260       |        Field Count = 3        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |     Scope Field Count = 2     |0|  MIBObjectIdentifierMarker  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |   Scope Field 1 Length = 1    |Index Count = 0|MIB OID Len=22 |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |      MIB Object Identifier = "1.3.6.1.2.1.4.31.3.1.1" ...     |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |           ...  MIB Object Identifier continued ...            |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |           ...  MIB Object Identifier continued ...            |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |           ...  MIB Object Identifier continued ...            |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |           ...  MIB Object Identifier continued ...            |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |MIB Object Identifier continued|0|  MIBObjectIdentifierMarker  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
    |   Scope Field 2 Length = 2    |Index Count = 0|MIB OID Len=22 |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |      MIB Object Identifier = "1.3.6.1.2.1.4.31.3.1.2" ...     |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |            ... MIB Object Identifier continued ...            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |            ... MIB Object Identifier continued ...            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |            ... MIB Object Identifier continued ...            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |            ... MIB Object Identifier continued ...            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |MIB Object Identifier continued|0|  MIBObjectIdentifierMarker  |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |       Field Length = 4        |Index Count = 2|MIB OID Len=23 |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |     MIB Object Identifier = "1.3.6.1.2.1.4.31.3.1.12" ...     |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |            ... MIB Object Identifier continued ...            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |            ... MIB Object Identifier continued ...            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |            ... MIB Object Identifier continued ...            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |            ... MIB Object Identifier continued ...            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |            ... MIB Object Identifier continued|0|MIB OID IE...|
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |... MIB OID IE | 1.3.6.1.2.1.4.31.3.1.1 Length |MIB OID Len=22 |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |     MIB Object Identifier = "1.3.6.1.2.1.4.31.3.1.1" ...      |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |            ... MIB Object Identifier continued ...            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |            ... MIB Object Identifier continued ...            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |            ... MIB Object Identifier continued ...            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |            ... MIB Object Identifier continued ...            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |   ... MIB Object Identifier   |0|         MIB OID IE          |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    | 1.3.6.1.2.1.4.31.3.1.2 Length | MIB OID Len=22| MIB Obj ID ...|
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |     MIB Object Identifier = "1.3.6.1.2.1.4.31.3.1.2" ...      |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |            ... MIB Object Identifier continued ...            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

Figure 17: Example of an Options Template for an Indexed MIB Object with two indices.

6.6. Indexed MIB Object with an IPFIX Information Element: Output Interface Queue Size in PSAMP Packet Report

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

The exported data looks like:

SRC ADDR	DST ADDR	PAK LEN	OUTPUT I/F	OUTPUT Q. LEN (ifOutQLen)
192.0.2.1	192.0.2.3	150	Eth 1/0 (15)	45
192.0.2.4	192.0.2.9	350	Eth 1/0 (15)	45
192.0.2.3	192.0.2.9	650	Eth 1/0 (15)	23
192.0.2.4	192.0.2.6	350	Eth 1/1 (16)	0

Table 3: Packet Report with Interface Output Queue Length (ifOutQLen) Data

The MIB object for the Interface Output Queue Length, ifOutQLen ("1.3.6.1.2.1.2.2.1.21"), is indexed by the ifIndex interface index as detailed in the IF-MIB [RFC2863]. If, for example, the interface index of "Eth 1/0" in the example is 15, the full MIB Object Identifier for (ifOutQLen) would be "1.3.6.1.2.1.2.2.1.21.15". Without a method to specify the index the full MIB OID would have to be used, which would mean specifying a new Template Record. Rather than export a separate Template Record for each Interface Index, it is more practical to identify the index in the Data Record itself.

In fact, only how the indexed object was indexed is necessary, although it is often useful to specify the index value. The example identifies the Egress Interface, but for other uses it may be sufficient to know that the ifOutQLen value was taken for the interface that the packet was switched out of, without identifying the actual interface.

The Template Record for the example Data Record contains the following Information Elements:

1. sourceIPv4Address

2. destinationIPv4Address

3. totalLengthIPv4

4. egressInterface

5. ifOutQLen indexed by: egressInterface

Figure 18 shows the exported Template Set detailing the Template for exporting a PSAMP Report with Interface Output Queue Length (ifOutQLen) (see Section 6.4).

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |          Set ID = TBD1        |          Length = 54          |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |        Template ID = 261      |        Field Count = 5        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0|   IE = sourceIPv4Address    |        Field Length = 4       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0| IE = destinationIPv4Address |        Field Length = 4       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0|    IE = totalLengthIPv4     |        Field Length = 4       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0|    IE = egressInterface     |        Field Length = 4       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0|IE=MIBObjectIdentifierMarker |        Field Length 4         |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    | Index Count=1 |MIB OID Len=20 |    MIB Object Identifier ...  |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |      ... MIB Object Identifier = "1.3.6.1.2.1.2.2.1.21"       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |           ...  MIB Object Identifier continued ...            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |           ...  MIB Object Identifier continued ...            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |           ...  MIB Object Identifier continued ...            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    | ... MIB OID continued         |0|    IE = egressInterface     |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  egressInterface Length = 4   |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 18: Example of Template for a PSAMP Report with ifOutQLen indexed by egressInterface

The corresponding IPFIX Data Record is shown in Figure 19. For the sake of the example, the interface index of "Eth 1/0" is 15 and the interface index of "Eth 1/1" is 16.

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |          Set ID = 261         |         Length = 84           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                          192.0.2.1                            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                          192.0.2.3                            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                             150                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                        15 (Eth 1/0)                           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                              45                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                          192.0.2.4                            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                          192.0.2.9                            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                             350                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                        15 (Eth 1/0)                           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                              45                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                          192.0.2.3                            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                          192.0.2.9                            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                             650                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                        15 (Eth 1/0)                           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                              23                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                          192.0.2.4                            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                          192.0.2.6                            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                             350                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                        16 (Eth 1/1)                           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                               0                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 19: Example of PSAMP Packet Report with ifOutQLen indexed by egressInterface

6.7. Indexed MIB Objects with a mix of MIB OID and IPFIX Information Element

TODO.

6.8. Using MIB Objects as IPFIX Options Scope fields

Both indexed and non-indexed MIB Objects may be used as IPFIX Options Scope fields as discussed in Section 5.2.

6.8.1. Using non-Indexed MIB Objects as Option Scope fields

In this example, a Cisco Telepresence system uses an IPFIX option to report bandwidth usage statistics. The ctpcLocalAddrType and ctpcLocalAddr OIDs from the CISCO-TELEPRESENCE-CALL MIB are used as scope fields to identify the Telepresence system. The ctpcLocalAddrType is expressed with a fixed size of 1 octet, while the ctpcLocalAddr is expressed using a variable length field.

These scope fields are followed by two non-scope fields containing the number of packets and bytes. IPFIX reduced size encoding is used to express each of these fields in 32 bits.

Therefore the Options Template Record for the example Data Record contains the following Information Elements:

1. ctpcLocalAddrType (1.3.6.1.4.1.9.9.644.1.2.1) (scope field)

2. ctpcLocalAddr (1.3.6.1.4.1.9.9.644.1.2.2) (scope field)

3. octetDeltaCount (non-scope field)

4. packetDeltaCount (non-scope field)

The IPFIX Options Template Record is shown in Figure 20.

```
     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 = TBD2         |          Length = 80          |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |       Template ID = 262       |        Field Count = 4        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |     Scope Field Count = 2     |0|  MIBObjectIdentifierMarker  |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |   Scope Field 1 Length = 1    |Index Count = 0|MIB OID Len=25 |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |    MIB Object Identifier = "1.3.6.1.4.1.9.9.644.1.2.1" ...    |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |           ...  MIB Object Identifier continued ...            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |           ...  MIB Object Identifier continued ...            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |           ...  MIB Object Identifier continued ...            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |           ...  MIB Object Identifier continued ...            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |           ...  MIB Object Identifier continued ...            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |... MIB OID ID |0|  MIBObjectIdentifierMarker | Scope Field ...|
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |...Length=65535|Index Count = 0|MIB OID Len=25 | MIB OID ID ...|
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  ... MIB Object Identifier = "1.3.6.1.4.1.9.9.644.1.2.2" ...  |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |           ...  MIB Object Identifier continued ...            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |           ...  MIB Object Identifier continued ...            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |           ...  MIB Object Identifier continued ...            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |           ...  MIB Object Identifier continued ...            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |           ...  MIB Object Identifier continued                |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0|     octetDeltaCount = 1     |       Field Length = 4        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0|    packetDeltaCount = 2     |       Field Length = 4        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 20: Example of an IPFIX Options Template Record using non-Indexed MIB Objects as scope fields

The corresponding IPFIX Options Data Record is shown in Figure 21.

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |         Set ID = 262          |          Length = 18          |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    | AddrType = 1  |  Length = 4   |  ctpcLocalAddrsystemID = ...  |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |         ... 192.0.2.1         |  octetDeltaCount = nnnn ...   |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    | ... octetDeltaCount continued |  packetDeltaCount = nnnn ...  |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    | ... packetDeltaCount continued|
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 21: Example of an IPFIX Options Data Record using non-Indexed MIB Objects as scope fields

6.8.2. Using Indexed MIB Objects as Option Scope fields

In this example, interface statistics are reported using ifName and ifInOctets from the IF-MIB [RFC2863]. Both of these fields are indexed by the ifIndex. The ifName and ifIndex are scope fields.

Therefore the Options Template Record for the example Data Record contains the following Information Elements:

1. ifName (1.3.6.1.2.1.31.1.1.1.1) (scope field) indexed by ifIndex

2. ifIndex (1.3.6.1.2.1.2.2.1.1) (scope field)

3. ifInOctets (1.3.6.1.2.1.2.2.1.10) (non-scope field) indexed by ifIndex

The IPFIX Options Template Record is shown in Figure 22.

```
     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 = TBD2         |          Length = 137         |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |       Template ID 263         |        Field Count = 3        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |     Scope Field Count = 2     |0|  MIBObjectIdentifierMarker  |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  Scope Field 1 Length = 65535 |Index Count = 1|MIB OID Len=22 |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |       MIB Object Identifier = "1.3.6.1.2.1.31.1.1.1.1"        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
|            ...  MIB Object Identifier continued ...           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            ...  MIB Object Identifier continued ...           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            ...  MIB Object Identifier continued ...           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            ...  MIB Object Identifier continued ...           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|MIB Object Identifier continued|0|  MIBObjectIdentifierMarker  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Scope Field 1 index Length = 4|MIB OID Len=19 | MIB OID ID ...|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      ... MIB Object Identifier = "1.3.6.1.2.1.2.2.1.1" ...     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            ...  MIB Object Identifier continued ...           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            ...  MIB Object Identifier continued ...           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            ...  MIB Object Identifier continued               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   ... MIB Object Identifier   |0|  MIBObjectIdentifierMarker  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Scope Field 2 Length = 4   |Index Count = 0|MIB OID Len=19 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       MIB Object Identifier = "1.3.6.1.2.1.2.2.1.1" ...       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            ...  MIB Object Identifier continued ...           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            ...  MIB Object Identifier continued ...           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            ...  MIB Object Identifier continued ...           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         ... MIB Obj Identifier continued      |0| MIBObject...|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|...Ident Marker|       Field Length =  4       |Index Count = 1|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|MIB OID Len=20 |MIB Object Identifier="1.3.6.1.2.1.2.2.1.10"...|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            ...  MIB Object Identifier continued ...           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            ...  MIB Object Identifier continued ...           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            ...  MIB Object Identifier continued ...           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            ...  MIB Object Identifier continued ...           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  ... MIB OID  |0|  MIBObjectIdentifierMarker  |  Field 1 ...  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
     |... index Len=4|MIB OID Len=19 |   MIB Object Identifier ...   |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |     ... MIB Object Identifier = "1.3.6.1.2.1.2.2.1.1" ...     |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |           ...  MIB Object Identifier continued ...            |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |           ...  MIB Object Identifier continued ...            |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |           ...  MIB Object Identifier continued ...            |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |... MIB Obj Id |
     +-+-+-+-+-+-+-+-+
```

Figure 22: Example of an IPFIX Options Template Record using Indexed MIB Objects as scope fields

The corresponding IPFIX Options Data Record is shown in Figure 23. For the sake of the example, the interface index of "Eth 1/1" is 15 and the ifInOctets are 1000.

```
      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 = 263          |          Length = 20          |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |  Length = 7   |             ifName = "Eth 1/1" ...            |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                      ... ifName continued                     |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                          ifIndex = 15                         |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                        ifInOctets = 1000                      |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 23: Example of an IPFIX Options Data Record using Indexed MIB Objects as scope fields

6.9. Using MIB Objects with IPFIX Structured Data

It’s possible to export both indexed and non-indexed MIB Objects using IPFIX Structured Data per [RFC6313] as shown in the example below.

TODO: insert example.

7. Configuration Considerations

When configuring a MIB OID for export, consideration should be given to whether the SNMP Context String should also be configurable. If a non-default Context String is used then it should be associated with the fields as per Section 5.5.

8. The Collecting Process's Side

This section describes the Collecting Process when using SCTP and PR-SCTP as the transport protocol. Any necessary changes to the Collecting Process specifically related to TCP or UDP transport protocols are specified in section 10 of [RFC5101].

The specifications in section 9 of [RFC5101] also apply to Collector's that implement this specification. In addition, the following specifications should be noted.

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

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

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

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

9. Applicability

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

Some monitoring applications periodically export an interface id to interface name mapping using IPFIX Options Templates. This could be expanded to include the MIB object "ifInUcastPkts" of the IF-MIB [RFC2863] indexed using the ingressInterface Information Element, as a index. This would give the input statistics for each interface which can be compared to the flow information to ensure the sampling rate is expected. Or, if there is no sampling, to ensure that all the expected packets are being monitored.

10. Security Considerations

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

The access to MIB objects is controlled by the configuration of the IPFIX exporter. This is consistent with the way IPFIX controls access to other Information Elements in general. The configuration of an IPFIX exporter determines which MIB objects are included in IPFIX flow records sent to certain collectors. Network operators should take care that only MIB objects are included in IPFIX flow records that the receiving flow collector is allowed to receive.

11. IANA Considerations

IPFIX Messages use two fields with assigned values. These are the IPFIX Version Number, indicating which version of the IPFIX Protocol was used to export an IPFIX Message, and the IPFIX Set ID, indicating the type for each set of information within an IPFIX Message.

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

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

12. References

12.1. Normative References

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

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

[RFC2578] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.

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

[RFC4293] Routhier, S., "Management Information Base for the Internet Protocol (IP)", RFC 4293, April 2006.

[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.

12.2. Informative References

[RFC2982] Kavasseri, R., "Distributed Management Expression MIB", RFC 2982, October 2000.

[RFC3444] Pras, A. and J. Schoenwaelder, "On the Difference between Information Models and Data Models", RFC 3444, January 2003.

[RFC4022] Raghunarayan, R., "Management Information Base for the Transmission Control Protocol (TCP)", RFC 4022, March 2005.

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

Authors' Addresses

Andrew Johnson
Cisco System, Inc.
96 Commercial Quay
Commercial Street
Edinburgh,   EH6 6LX
UK

Phone: +44 131 561 3641
Email: andrjohn@cisco.com

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

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

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

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

Juergen Schoenwaelder
Jacobs University Bremen
Campus Ring 1
Bremen,   28725
Germany

Phone: +49 421 200-3587
Email: j.schoenwaelder@jacobs-university.de

IP Flow Information Export
Internet-Draft
Intended status: Standards Track
Expires: March 17, 2012
S. Kashima
K. Nakata
NTT
A. Kobayashi
NTT East
September 14, 2011

Information Elements for Data Link Layer Traffic Measurement
draft-kashima-ipfix-data-link-layer-monitoring-06

Abstract

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

Status of this Memo

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

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

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

This Internet-Draft will expire on March 17, 2012.

Copyright Notice

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

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

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

Table of Contents

1. Introduction

Ethernet [IEEE802.1D] and VLAN (Virtual LAN) [IEEE802.1Q-2005] technologies used to be used only in Local Area Networks. Recently, they have been used in Wide Area Networks, e.g., L2-VPN services. Accordingly, the IEEE802.1Q standard has been enhanced to IEEE802.1ad [IEEE802.1ad-2005] and IEEE802.1ah [IEEE802.1ah-2008]. And, Ethernet in data center also has been enhanced for server virtualization and I/O consolidation.

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

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

1.1. Conventions Used in This Document

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

2. Extended Ethernet Technology

2.1. Wide-Area Ethernet Summary

Provider Bridge [IEEE802.1ad-2005] and Provider Backbone Bridge [IEEE802.1ah-2008], which are standards for the Wide-Area Ethernet, are described below.

- o In Provider Bridge [IEEE802.1ad-2005], there are two VLAN IDs: Service VLAN Identifier (S-VID) and Customer VLAN Identifier (C-VID). S-VID is assigned to an Ethernet frame by a service provider, while C-VID is independently assigned to an Ethernet frame by a customer. Frame switching in a service provider network is based on only S-VID.
- o In Provider Backbone Bridge [IEEE802.1ah-2008], new Ethernet fields, such as Backbone VLAN Identifier (B-VID) and Backbone Service Instance Identifier (I-SID), are introduced to overcome the limitations on the VLAN identifier space on IEEE802.1ad

[IEEE802.1ad-2005] and to isolate the service provider and customer identifier spaces. Frame switching is based on a 12-bit B-VID, and customer identification is based on a 24-bit I-SID. A flexible network design has become possible because network management is separated from customer management. Other Ethernet fields that indicate quality of service (QoS) class are B-PCP, B-DEI, I-PCP, and I-DEI.

Provider Backbone Bridge enables a wide-area Ethernet service to be improved from a flat network to a hierarchical network co-existing Provider Bridge and Provider Backbone Bridge.

Frame formats used in Wide-Area Ethernet are shown in Appendix A.

2.2. Data Center Bridging Summary

In data center networks, Ethernet needs to be enhanced to provide the flexibility, mobility for server virtualization, and I/O consolidation. In IEEE802.1 Data Center Bridging Task Group, several Ethernet header formats are proposed to enable a simplifying networks and server managements.

The one of the enhanced methods is Bridge Port Extension [IEEE802.1Qbh], which brings a traffic exchange point to upper bridges. Bridge Port Extension introduces a Ethernet format named Extension Tag (E-TAG) in addition to existing Service VLAN Tag (S-TAG) and Customer VLAN Tag (C-TAG) to move the policy enhancement to upper bridges in data center network. On the other hand, the complexity for traffic measurement would be increased, because multiple Ethernet header formats as shown in Appendix B co-exist in the same network.

2.3. Multiple Path Ethernet Summary

Inside data center networks, Ethernet need to be enhaced to support multiple path for effective utilization of bandwidth. Existing Ethernet does not support multiple path because it is based on STP (Spanning Tree Protocol) [IEEE802.1D].

Two solutions for multiple path Ethernet are discussed for standardization. One is Shortest Path Bridging [IEEE802.1aq] in IEEE 802, the other is TRILL (Transparent Interconnection of Lots of Links) [RFC6325] in IETF. Both solutions realize multiple path with IS-IS [ISO_IEC.10589_2002] and Layer 2 over Layer 2 encapsulation. Shortest Path Bridging uses IEEE802.1ah [IEEE802.1ah-2008] header as an encapsulation technology and TRILL uses an unique header as shown in Section 3 of [RFC6325].

In multiple path Ehternet network, a traffic measurement based on MAC address and VLAN Tag is more important than single path Ethernet network in which traffic measurement is based on only VLAN Tag.

2.4. VXLAN Summary

Inside and between data center networks, the existing Ethernet and VLAN technologies have two problems. One is that 12 bit of VLAN ID does not have enough length for identifying a lot of tenants. The other is that it is difficult to place same VLAN in multiple data centers.

The solution for the problems is VXLAN [I-D.mahalingam-dutt-dcops-vxlan]. VXLAN has 24 bit length of identifier named VXLAN Segment ID/VXLAN Network Identifier (VNI) and uses Layer 2 over Layer 3 encapsulation with the frame format as show in Section 5 of [I-D.mahalingam-dutt-dcops-vxlan].

3. New Information Elements

The following Information Elements are necessary for enabling the IPFIX/PSAMP traffic measurement for data link layer, which is not limited to Ethernet because the method can be applied to other data link protocols as well. Note that these are proposed IDs, subject to approval by IANA.

ID	Name
312	dataLinkFrameSize
315	dataLinkFrameSection
347	dataLinkFrameType
348	sectionOffset
349	sectionObservedOctets
350	dot1qServiceInstanceTag
351	dot1qServiceInstanceId
352	dot1qServiceInstancePriority
353	dot1qCustomerDestinationMacAddress
354	dot1qCustomerSourceMacAddress

3.1. dataLinkFrameSize

Description:

This Information Element specifies the length of the selected data link frame.

The data link layer is defined in [ISO_IEC.7498-1_1994].

Abstract Data Type: unsigned16

Data Type Semantics: quantity

ElementId: 312

Status: current

3.2. dataLinkFrameSection

Description:

This Information Element carries n octets from the data link frame of a selected frame, starting sectionOffset octets into the frame.

The sectionObservedOctets expresses how much data was observed, while the remainder is padding.

When the sectionObservedOctets field corresponding to this Information Element exists, this Information Element MAY have a fixed length and MAY be padded, or MAY have a variable length.

When the sectionObservedOctets field corresponding to this Information Element does not exist, this Information Element SHOULD have a variable length and MUST NOT be padded. In this case, the size of the exported section may be constrained due to limitations in the IPFIX protocol.

Further Information Elements, i.e., dataLinkFrameType and dataLinkFrameSize are needed to specify the data link type and the size of the data link frame of this Information Element. A set of these Information Elements MAY be contained in a structured data type, as expressed in [RFC6313]. Or a set of these Information Elements MAY be contained in one Flow Record as shown in Appendix C.

The data link layer is defined in [ISO_IEC.7498-1_1994].

Abstract Data Type: octetArray

ElementId: 315

Status: current

3.3. dataLinkFrameType

Description:

This Information Element specifies the type of the selected data link frame.

The following data link types are defined here.

- 0x01 ETHERNET

Further values may be assigned by IANA.

The data link layer is defined in [ISO_IEC.7498-1_1994].

Abstract Data Type: unsigned16

Data Type Semantics: identifier

ElementId: 347

Status: current

3.4. sectionOffset

Description:

This Information Element specifies the offset of the packet section (e.g., dataLinkFrameSection, ipHeaderPacketSection, ipPayloadPacketSection, mplsLabelStackSection and mplsPayloadPacketSection). If this Information Element is omitted, it defaults to zero.

Abstract Data Type: unsigned16

Data Type Semantics: quantity

ElementId: 348

Status: current

3.5. sectionObservedOctets

Description:

This Information Element specifies the observed length of the packet section (e.g., dataLinkFrameSection, ipHeaderPacketSection, ipPayloadPacketSection, mplsLabelStackSection and mplsPayloadPacketSection) when padding is used.

The packet section may be of a fixed size larger than the sectionObservedOctets. In this case, octets in the packet section beyond the sectionObservedOctets MUST follow the [RFC5101] rules for padding (ie, be composed of zero (0) valued octets).

Abstract Data Type: unsigned16

Data Type Semantics: quantity

ElementId: 349

Status: current

3.6. dot1qServiceInstanceTag

Description:

This Information Element, which may have 16 octets length, carries the Backbone Service Instance Tag (I-TAG) Tag Control Information (TCI) field of an Ethernet frame as described in [IEEE802.1ah-2008].

Abstract Data Type: octetArray

Data Type Semantics: identifier

ElementId: 350

Status: current

3.7. dot1qServiceInstanceId

Description:

The value of the 3-bit Backbone Service Instance Priority Code Point (I-PCP) portion of the Backbone Service Instance Tag (I-TAG) Tag Control Information (TCI) field of an Ethernet frame as described in section 9.8 of [IEEE802.1ah-2008]. The structure and semantics within the Tag Control Information field are defined in IEEE802.1ah.

Abstract Data Type: unsigned32

Data Type Semantics: identifier

ElementId: 351

Status: current

3.8. dot1qServiceInstancePriority

Description:

The value of the 24-bit Backbone Service Instance Identifier (I-SID) portion of the Backbone Service Instance Tag (I-TAG) Tag Control Information (TCI) field of an Ethernet frame as described in section 9.8 of [IEEE802.1ah-2008]. The structure and semantics within the Tag Control Information field are defined in IEEE802.1ah.

Abstract Data Type: unsigned8

Data Type Semantics: identifier

ElementId: 352

Status: current

3.9. dot1qCustomerDestinationMacAddress

Description:

The value of the Customer Destination Address (C-DA) portion of the Backbone Service Instance Tag (I-TAG) Tag Control Information (TCI) field of an Ethernet frame as described in section 9.8 of [IEEE802.1ah-2008]. The structure and semantics within the Tag Control Information field are defined in IEEE802.1ah.

Abstract Data Type: macAddress

Data Type Semantics: identifier

ElementId: 353

Status: current

3.10. dot1qCustomerSourceMacAddress

Description:

The value of the Customer Source Address (C-SA) portion of the Backbone Service Instance Tag (I-TAG) Tag Control Information (TCI) field of an Ethernet frame as described in section 9.8 of [IEEE802.1ah-2008]. The structure and semantics within the Tag Control Information field are defined in IEEE802.1ah.

Abstract Data Type: macAddress

Data Type Semantics: identifier

ElementId: 354

Status: current

4. Modification of Existing Information Elements Related to Packet Section

--- This is open issue. ---

New Information Elements related to packet section (ie, sectionOffset and sectionObservedOctets) can be applied to not only dataLinkFrameSection but also all kinds of packet section. In this case, existing Information Elements Description should be modified as follows:

4.1. ipHeaderPacketSection

Description:

TBD

Abstract Data Type: octetArray

ElementId: 313

Status: current

4.2. ipPayloadPacketSection

Description:

TBD

Abstract Data Type: octetArray

ElementId: 314

Status: current

4.3. mplsLabelStackSection

Description:

TBD

Abstract Data Type: octetArray

ElementId: 316

Status: current

4.4. mplsPayloadPacketSection

Description:

TBD

Abstract Data Type: octetArray

ElementId: 317

Status: current

5. Modification of Existing Information Elements Related to VLAN Tag

Information Elements related to Backbone Service Instance Tag (I-TAG) and Backbone VLAN Tag (B-TAG) are required in order to monitor IEEE802.1ah traffic with IPFIX/PSAMP. Because I-TAG has different structure and semantics from Service VLAN Tag (S-TAG) and Customer VLAN Tag (C-TAG), new Information Elements related to I-TAG are added in section 3. But because B-TAG has same different structure and semantics with C-TAG and S-TAG, Information Elements related to B-TAG reuse existing Information Elements related to C-TAG and S-TAG. Though they reuse existing Information Elements, it required to modify existing Descriptions and Reference as follows:

5.1. dot1qVlanId

Description:

The value of the 12-bit VLAN Identifier portion of the Tag Control Information field of an Ethernet frame as described in section 3.5.5 of [IEEE802.3-2005]. The structure and semantics within the Tag Control Information field are defined in IEEE P802.1Q. In case of a QinQ frame it represents the outer tag’s VLAN identifier, in case of an IEEE 802.1ad frame it represents the Service VLAN identifier in the S-TAG Tag Control Information (TCI) field as described in [IEEE802.1ad-2005] and in case of an IEEE 802.1ah frame it represents the Backbone VLAN identifier in the B-TAG Tag Control Information (TCI) field as described in

[IEEE802.1ah-2008].

Abstract Data Type: unsigned16

Data Type Semantics: identifier

ElementId: 243

Status: current

Reference:

(1) [IEEE802.3-2005]
(2) [IEEE802.1ad-2005]
(3) [IEEE802.1ah-2008]

5.2. dot1qPriority

Description:

The value of the 3-bit User Priority portion of the Tag Control Information field of an Ethernet frame as described in section 3.5.5 of [IEEE802.3-2005]. The structure and semantics within the Tag Control Information field are defined in IEEE P802.1Q. In case of a QinQ frame it represents the outer tag’s 3-bit Class of Service (CoS) identifier, in case of an IEEE 802.1ad frame it represents the 3-bit Priority Code Point (PCP) portion of the S-TAG Tag Control Information (TCI) field as described in [IEEE802.1ad-2005] and in case of an IEEE 802.1ah frame it represents the 3-bit Priority Code Point (PCP) portion of the B-TAG Tag Control Information (TCI) field as described in [IEEE802.1ah-2008].

Abstract Data Type: unsigned8

Data Type Semantics: identifier

ElementId: 244

Status: current

Reference:

(1) [IEEE802.3-2005]

(2) [IEEE802.1ad-2005]
(3) [IEEE802.1ah-2008]

5.3. dot1qCustomerVlanId

Description:

In case of a QinQ frame it represents the inner tag’s (*) VLAN identifier, in case of an IEEE 802.1ad frame it represents the Customer VLAN identifier in the C-TAG Tag Control Information (TCI) field as described in [IEEE802.1ad-2005] and in case of an IEEE 802.1ah frame it represents the Customer VLAN identifier in the C-TAG Tag Control Information (TCI) field in encapsulated IEEE 802.1ad or IEEE 802.1Q frame as described in [IEEE802.1ah-2008].
(*) Note: the 802.1Q tag directly following the outer one.

Abstract Data Type: unsigned16

Data Type Semantics: identifier

ElementId: 245

Status: current

Reference:

(1) [IEEE802.1ad-2005]
(2) [IEEE802.1Q-2005]
(3) [IEEE802.1ah-2008]

5.4. dot1qCustomerPriority

Description:

In case of a QinQ frame it represents the inner tag’s (*) Class of Service (CoS) identifier, in case of an IEEE 802.1ad frame it represents the 3-bit Priority Code Point (PCP) portion of the C-TAG Tag Control Information (TCI) field as described in [IEEE802.1ad-2005] and in case of an IEEE 802.1ah frame it represents the 3-bit Priority Code Point (PCP) portion of the C-TAG Tag Control Information (TCI) field in encapsulated IEEE 802.1ad or IEEE 802.1Q frame as described in [IEEE802.1ad-2005].
(*) Note: the 802.1Q tag directly following the outer one.

Abstract Data Type: unsigned8

Data Type Semantics: identifier

ElementId: 246

Status: current

Reference:

(1) [IEEE802.1ad-2005]
(2) [IEEE802.1Q-2005]
(3) [IEEE802.1ah-2008]

6. Security Considerations

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

7. IANA Considerations

This document requests that the Information Element IDs are allocated as shown in section 3

In addition, the dataLinkFrameType Information Element requires the creation of new IANA registries.

And this document requests that the existing Information Element Description are modified as shown in section 4 and 5

8. References

8.1. Normative References

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

8.2. Informative References

[I-D.mahalingam-dutt-dcops-vxlan]
Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "VXLAN: A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", draft-mahalingam-dutt-dcops-vxlan-00 (work in progress), August 2011.

[IEEE802.1D]
IEEE Computer Society, "IEEE Standards for Local and

Metropolitan Area Networks: Media Access Control (MAC) Bridges", IEEE Std 802.1D-2004, June 2004.

[IEEE802.1Q-2005]
IEEE Computer Society, "IEEE Standards for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks", IEEE Std 802.1Q-2005, May 2006.

[IEEE802.1Qbh]
IEEE Computer Society, "Draft Standards for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks Amendment: Bridge Port Extension", IEEE Std P802.1Qbh/D1.1, February 2011.

[IEEE802.1ad-2005]
IEEE Computer Society, "IEEE Standards for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks Amendment 4: Provider Bridges", IEEE Std 802.1ad-2005, May 2006.

[IEEE802.1ah-2008]
IEEE Computer Society, "IEEE Standards for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks Amendment 7: Provider Backbone Bridges", IEEE Std 802.1ah-2008, August 2008.

[IEEE802.1aq]
IEEE Computer Society, "Draft Standards for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks Amendment: Shortest Path Bridging", IEEE Std P802.1aq/D3.6, February 2011.

[IEEE802.3-2005]
IEEE Computer Society, "IEEE Standard for 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 Std 802.3-2005, December 2005.

[ISO_IEC.10589_2002]
International Organization for Standardization, "Information technology -- Telecommunications and information exchange between systems -- Intermediate System to Intermediate System intra-domain routing information exchange protocol for use in conjunction with the protocol for providing the connectionless-mode network

service (ISO 8473)", ISO Standard 10589:2002, November 2002.

[ISO_IEC.7498-1_1994]
International Organization for Standardization, "Information technology -- Open Systems Interconnection -- Basic Reference Model: The Basic Mode", ISO Standard 7498-1:1994, June 1996.

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

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

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

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

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

[RFC6313] Claise, B., Dhandapani, G., Aitken, P., and S. Yates, "Export of Structured Data in IP Flow Information Export (IPFIX)", RFC 6313, July 2011.

[RFC6325] Perlman, R., Eastlake, D., Dutt, D., Gai, S., and A. Ghanwani, "Routing Bridges (RBridges): Base Protocol Specification", RFC 6325, July 2011.

Appendix A. Frame Formats in Wide-Area Ethernet Network

```
   0                   1                   2                   3
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                              C-DA                             |
   +                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                               |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
   |                              C-SA                             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          Length/Type          |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
   |                                                               |
   ~                         Customer Data                         ~
   ~                                                               ~
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure A-1: IEEE802.1D Frame Format in Customer Bridged Network

```
   0                   1                   2                   3
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                              C-DA                             |
   +                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                               |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
   |                              C-SA                             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |        C-TAG TCI=0x8100       |C-PCP|C|         C-VID         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          Length/Type          |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
   |                                                               |
   ~                         Customer Data                         ~
   ~                                                               ~
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure A-2: IEEE802.1Q Frame Format in Customer Bridged Network

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              C-DA                             |
+                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                              C-SA                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        S-TAG TCI=0x88a8       |S-PCP|D|         S-VID         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Length/Type          |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                                                               |
~                         Customer Data                         ~
~                                                               ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure A-3: IEEE802.1ad (no C-Tag) Frame Format in Provider Bridged Network

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              C-DA                             |
+                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                              C-SA                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        S-TAG TCI=0x88a8       |S-PCP|D|         S-VID         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        C-TAG TCI=0x8100       |C-PCP|C|         C-VID         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Length/Type          |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                                                               |
~                         Customer Data                         ~
~                                                               ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure A-4: IEEE802.1ad (C-Tagged) Frame Format in Provider Bridged Network

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              B-DA                             |
+                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                              B-SA                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        B-TAG TCI=0x88a8       |B-PCP|D|         B-VID         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        I-TAG TCI=0x88e7       |I-PCP|D|U| Res |     I-SID     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             I-SID             |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                              C-DA                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              C-SA                             |
+                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |          Length/Type          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                         Customer Data                         ~
~                                                               ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure A-5: IEEE802.1ah (no C-Tag) Frame Format in Provider Backbone Bridged Network

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              B-DA                             |
+                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                              B-SA                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        B-TAG TCI=0x88a8       |B-PCP|D|         B-VID         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        I-TAG TCI=0x88e7       |I-PCP|D|U| Res |     I-SID     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             I-SID             |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                              C-DA                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              C-SA                             |
+                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |        C-TAG TCI=0x8100       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|C-PCP|C|         C-VID         |          Length/Type          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                         Customer Data                         ~
~                                                               ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure A-6: IEEE802.1ah (C-Tagged) Frame Format in Provider Backbone Bridged Network

Appendix B. Frame Formats in Data Center Network

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              C-DA                             |
+                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                              C-SA                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        S-TAG TCI=0x88a8       |S-PCP|D|         S-VID         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Length/Type          |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                                                               |
~                         Customer Data                         ~
~                                                               ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure B-1: IEEE802.1Qbh (S-TAG) Frame Format in Data Center Network

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              C-DA                             |
+                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                              C-SA                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        S-TAG TCI=0x88a8       |S-PCP|D|         S-VID         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        C-TAG TCI=0x8100       |C-PCP|C|         C-VID         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Length/Type          |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                                                               |
~                         Customer Data                         ~
~                                                               ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure B-2: IEEE802.1Qbh (S-TAG+C-TAG) Frame Format in Data Center Network

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              C-DA                             |
+                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                              C-SA                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        E-TAG TCI=0xXXXX       |S-PCP|D|     Ingress_E-CID     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Res|           E-CID           |                Res            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Length/Type          |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                                                               |
~                         Customer Data                         ~
~                                                               ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure B-3: IEEE802.1Qbh (E-TAG) Frame Format in Data Center Network

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              C-DA                             |
+                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                              C-SA                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        E-TAG TCI=0xXXXX       |S-PCP|D|     Ingress_E-CID     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Res|           E-CID           |                Res            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        C-TAG TCI=0x8100       |C-PCP|C|         C-VID         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Length/Type          |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                                                               |
~                         Customer Data                         ~
~                                                               ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure B-4: IEEE802.1Qbh (E-TAG+C-TAG) Frame Format in Data Center

Network

Appendix C. Template Formats Example

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Set ID (0x0002)        |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Template ID (0x0103)     |      Field Count (0x0008)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   ingressInterface (0x000A)   |     Field Length (0x0004)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   egressInterface (0x000E)    |     Field Length (0x0004)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|observationTimeSeconds (0x0142)|     Field Length (0x0008)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   dataLinkFrameSize (0x0138)  |     Field Length (0x0002)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| dataLinkFrameSection (0x013B) |     Field Length (0xFF40)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   dataLinkFrameType (0x015B)  |     Field Length (0x0002)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     sectionOffset (0x015C)    |     Field Length (0x0002)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|sectionObservedOctets (0x015D) |     Field Length (0x0002)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure C-1: Template Format Example

Authors' Addresses

Shingo Kashima
NTT Information Sharing Platform Lab.
Midori-Cho 3-9-11
Musashino-shi, Tokyo 180-8585
Japan

Phone: +81 422 59 3894
Email: kashima@nttv6.net

Kensuke Nakata
NTT Information Sharing Platform Lab.
Midori-Cho 3-9-11
Musashino-shi, Tokyo 180-8585
Japan

Phone: +81 422 59 6958
Email: nakata@nttv6.net

Atsushi Kobayashi
NTT EAST IT Innovation Department
25F 3-20-2 Nishi-shinjuku
Shinjuku-ku, Tokyo 163-1425
Japan

Phone: +81-3-5353-3636
Email: akoba@nttv6.net

IPFIX Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 29, 2012

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

September 26, 2011

Flow Aggregation for the IP Flow Information Export (IPFIX) Protocol
draft-trammell-ipfix-a9n-04.txt

Abstract

This document describes the export of aggregated Flow information using IPFIX. An Aggregated Flow is essentially an IPFIX Flow representing packets from multiple Original Flows sharing some set of common properties. The document describes Aggregated Flow export within the framework of IPFIX Mediators and defines an interoperable, implementation-independent method for Aggregated Flow export.

Status of this Memo

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

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

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

This Internet-Draft will expire on March 29, 2012.

Copyright Notice

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

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

Table of Contents

1. Introduction

The assembly of packet data into Flows serves a variety of different purposes, as noted in the requirements [RFC3917] and applicability statement [RFC5472] for the IP Flow Information Export (IPFIX) protocol [RFC5101]. Aggregation beyond the flow level, into records representing multiple Flows, is a common analysis and data reduction technique as well, with applicability to large-scale network data analysis, archiving, and inter-organization exchange. This applicability in large-scale situations, in particular, led to the inclusion of aggregation as part of the IPFIX Mediators Problem Statement [RFC5982], and the definition of an Intermediate Aggregation Process in the Mediator framework [RFC6183].

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

Aggregation as defined and described in this document covers the applications defined in [RFC5982], including 5.1 "Adjusting Flow Granularity", 5.4 "Time Composition", and 5.5 "Spatial Composition". However, this document specifies a more flexible architecture for an Intermediate Aggregation Process than that envisioned by the original Mediator work, in Section 4.2. Instead of a focus on these specific limited use cases, the Intermediate Aggregation Process is specified to cover any activity commonly described as "flow aggregation".

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

Since Aggregated Flows as defined in the following section are essentially Flows, the IPFIX protocol [RFC5101] can be used to export, and the IPFIX File Format [RFC5655] can be used to store, aggregated data "as-is"; there are no changes necessary to the protocol. This document provides a common basis for the application of IPFIX to the handling of aggregated data, through a detailed terminology, Intermediate Aggregation Process architecture, and methods for Original Flow counting and counter distribution across intervals.

1.1. IPFIX Protocol Overview

In the IPFIX protocol, { type, length, value } tuples are expressed in templates containing { type, length } pairs, specifying which { value } fields are present in data records conforming to the Template, giving great flexibility as to what data is transmitted. Since Templates are sent very infrequently compared with Data Records, this results in significant bandwidth savings. Various different data formats may be transmitted simply by sending new Templates specifying the { type, length } pairs for the new data format. See [RFC5101] for more information.

The IPFIX information model [RFC5102] defines a large number of standard Information Elements which provide the necessary { type } information for Templates. The use of standard elements enables interoperability among different vendors' implementations. Additionally, non-standard enterprise-specific elements may be defined for private use.

1.2. IPFIX Documents Overview

"Specification of the IPFIX Protocol for the Exchange of IP Traffic Flow Information" [RFC5101] and its associated documents define the IPFIX Protocol, which provides network engineers and administrators with access to IP traffic flow information.

"Architecture for IP Flow Information Export" [RFC5470] defines the architecture for the export of measured IP flow information out of an IPFIX Exporting Process to an IPFIX Collecting Process, and the basic terminology used to describe the elements of this architecture, per the requirements defined in "Requirements for IP Flow Information Export" [RFC3917]. The IPFIX Protocol document [RFC5101] then covers the details of the method for transporting IPFIX Data Records and Templates via a congestion-aware transport protocol from an IPFIX Exporting Process to an IPFIX Collecting Process.

This document specifies an Intermediate Process which may be applied at an IPFIX Mediator. "IP Flow Information Export (IPFIX) Mediation: Problem Statement" [RFC5982] introduces the concept of IPFIX Mediators, and defines the use cases for which they were designed; "IP Flow Information Export (IPFIX) Mediation: Framework" [RFC6183] then provides an architectural framework for Mediators. Protocol-level issues (e.g., template and observation domain handling across Mediators) are covered by "Specification of the Protocol for IPFIX Mediation" [I-D.claise-ipfix-mediation-protocol].

2. Terminology

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

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

In addition, this document defines the following terms

Aggregated Flow: A Flow, as defined by [RFC5101], derived from a set of zero or more original Flows within a defined Aggregation Interval. The two primary differences between a Flow and an Aggregated Flow in the general case are (1) that the time interval (i.e., the two-tuple of start and end times) of a Flow is derived from information about the timing of the packets comprising the Flow, while the time interval of an Aggregated Flow are externally imposed; and (2) that an Aggregated Flow may represent zero packets (i.e., an assertion that no packets were seen for a given Flow Key in a given time interval). Note that an Aggregated Flow is defined in the context of an Intermediate Aggregation Process only. Once an Aggregated Flow is exported, it is essentially a Flow as in [RFC5101] and can be treated as such.

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

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

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

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

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

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

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

3. Use Cases for IPFIX Aggregation

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

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

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

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

4. Architecture for Flow Aggregation

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

4.1. Aggregation within the IPFIX Architecture

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

```
   +===========================================+
   |  IPFIX Exporter        +----------------+ |
   |                        | Metering Proc. | |
   | +-----------------+    +----------------+ |
   | | Metering Proc.  | or |      IAP       | |
   | +-----------------+----+----------------+ |
   | |           Exporting Process           | |
   | +-|----------------------------------|--+ |
   +===|==================================|====+
       |                                  |
       |         (Aggregated Flow Export) |
       |                                  |
   +===|===========================+      |
   |   |  Mediator                 |      |
   + +-V-------------------+       |      |
   | | Collecting Process  |       |      |
   + +---------------------+       |      |
   | |         IAP         |       |      |
   + +---------------------+       |      |
   | |  Exporting Process  |       |      |
   + +-|-------------------+       |      |
   +===|===========================+      |
       |                                  |
       | (Aggregating Mediator)           |
       |                                  |
   +===|==================================|=====+
   |   | Collector                        |     |
   | +-V----------------------------------V-+   |
   | |         Collecting Process           |   |
   | +------------------+-------------------+   |
   |                    |       IAP         |   |
   |                    +-------------------+   |
   |                    |   File Writer     |   |
   |                    +----------|--------+   |
   +================================|===========+
                                    |
          (Aggregation for Storage) |
                                    V
                             +------------------+
                             |    IPFIX File    |
                             +------------------+
```

Figure 1: Potential Aggregation Locations

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

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

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

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

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

```
packets --+                     +- IPFIX Messages -+
          |                     |                  |
          V                     V                  V
+==================+ +====================+ +==============+
| Metering Process | | Collecting Process | | File Reader  |
|                  | +====================+ +==============+
| (original Flows  |            |  Original Flows  |
|  or direct aggr.)|            V                  V
+ - - - - - - - - -+======================================+
|           Intermediate Aggregation Process (IAP)        |
+=========================================================+
          | Aggregated                  Aggregated |
          | Flows                            Flows |
          V                                        V
+===================+                       +==============+
| Exporting Process |                       | File Writer  |
+===================+                       +==============+
          |                                        |
          +------------> IPFIX Messages <----------+
```

Figure 2: Data paths through the aggregation process

In the special case that aggregation is used together with correlation to aggregate the output from multiple Metering Processes, the arrangement would appear as in Figure 3; the details of correlation are, as noted above, out of scope for this document.

```
packets --+---------------------+------------------+
          |                     |                  |
          V                     V                  V
+====================+ +====================+ +====================+
| Metering Process 1 | | Metering Process 2 | | Metering Process n |
+====================+ +====================+ +====================+
          |                     |  Original Flows  |
          V                     V                  V
+==================================================================+
|                      Correlation process                         |
+------------------------------------------------------------------+
|                Intermediate Aggregation Process (IAP)            |
+==================================================================+
          | Aggregated                  Aggregated |
          | Flows                            Flows |
          V                                        V
+===================+                     +==============+
| Exporting Process |                     | File Writer |
+===================+                     +==============+
          |                                        |
          +------------> IPFIX Messages <----------+
```

Figure 3: Aggregation and correlation colocated

4.2. Intermediate Aggregation Process Architecture

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

```
           Original Flows
   +=============|============================================+
   |             |  Intermediate Aggregation Process (IAP)    |
   |             V                                            |
   |  +---------------------------------------------------+   |
   |  |            interval distribution (temporal)       |   |
   |  +---------------------------------------------------+   |
   |           | ^                    | ^             |       |
   |           | |  Partially Aggregated | |    Flows      |       |
   |           V |                    V |             |       |
   |  +-----------------+       +------------------+     |       |
   |  |  key aggregation  |<------|  value aggregation |     |       |
   |  |     (spatial)     |------>|      (spatial)     |     |       |
   |  +-----------------+       +------------------+     |       |
   |            |                    |                |       |
   |            |   Partially Aggregated  |      Flows      |       |
   |            V                    V                V       |
   |  +---------------------------------------------------+   |
   |  |                 aggregate combination              |   |
   |  +---------------------------------------------------+   |
   |                                 |                        |
   +=================================|========================+
                                     V
                              Aggregated Flows
```

Figure 4: Conceptual model of aggregation operations within an IAP

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

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

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

Section 5.3.

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

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

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

5. IP Flow Aggregation Operations

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

5.1. Temporal Aggregation through Interval Distribution

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

The length of the imposed interval itself has tradeoffs. Shorter intervals allow higher resolution aggregated data and, in streaming applications, faster reaction time. Longer intervals lead to greater data reduction and simplified counter distribution. Specifically, counter distribution is greatly simplified by the choice of an interval longer than the duration of longest Original Flow, itself generally determined by the Original Flow’s Metering Process active timeout; in this case an Original Flow can contribute to at most two Aggregated Flows, and the more complex value distribution methods become inapplicable.

```
|                |                |                |
| |<--Flow A-->| |                |                |
|       |<--Flow B-->|            |                |
|         |<-------------Flow C-------------->|    |
|                |                |                |
|   interval 0   |   interval 1   |   interval 2   |
```

Figure 5: Illustration of interval distribution

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

5.1.1. Distributing Values Across Intervals

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

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

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

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

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

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

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

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

Simulated Process: Each counter of the Original Flow is distributed among the intervals of the Aggregated Flows according to some function the Aggregation Process uses based upon properties of Flows presumed to be like the Original Flow. For example, Flow

Records representing bulk transfer might follow a more or less proportional uniform distribution, while interactive processes are far more bursty.

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

A method for exporting the distribution of counters across multiple Aggregated Flows is detailed in Section 7.4. In any case, counters MUST be distributed across the multiple Aggregated Flows in such a way that the total count is preserved, within the limits of accuracy of the implementation (e.g., inaccuracy introduced by the use of floating-point numbers is tolerable). This property allows data to be aggregated and re-aggregated without any loss of original count information. To avoid confusion in interpretation of the aggregated data, all the counters for a set of given Original Flows SHOULD be distributed via the same method.

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

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

5.1.2. Time Composition

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

5.2. Spatial Aggregation of Flow Keys

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

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

In any key aggregation operation, reduction and/or replacement may be applied any number of times in any order. Which of these operations are supported by a given implementation is implementation- and application-dependent. Key aggregation may aggregate Original Flows with different sets of Flow Keys; only the Flow Keys of the resulting Aggregated Flows of any given Key aggregation operation need to contain the same set of fields.

```
Original Flow Keys
+---------+--------+---------+---------+-------+-----+
| src ip4 | dst ip4 | src port | dst port | proto | tos |
+---------+--------+---------+---------+-------+-----+
     |         |         |          |         |      |
  retain   mask /24      X          X         X      X
     V         V
+---------+------------+
| src ip4 | dst ip4 /24 |
+---------+------------+
Aggregated Flow Keys (by source address and destination class-C)
```

Figure 6: Illustration of key aggregation by reduction

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

```
Original Flow Keys
+---------+--------+---------+---------+-------+-----+
| src ip4 | dst ip4 | src port | dst port | proto | tos |
+---------+--------+---------+---------+-------+-----+
     |         |         |          |         |      |
+-------------------+    X          X         X      X
| ASN lookup table  |
+-------------------+
     V         V
+---------+---------+
| src asn | dst asn |
+---------+---------+
Aggregated Flow Keys (by source and dest ASN)
```

Figure 7: Illustration of key aggregation by reduction and replacement

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

5.2.1. Counting Original Flows

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

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

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

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

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

SHOULD be configured to export Flows with equal or similar activeTimeout and inactiveTimeout configuration values, and the same cacheMode, as defined in section 4.3 of [I-D.ietf-ipfix-configuration-model].

5.2.2. Counting Distinct Key Values

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

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

5.3. Spatial Aggregation of Non-Key Fields

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

5.3.1. Counter Statistics

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

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

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

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

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

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

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

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

5.4. Aggregation Combination

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

6. Additional Considerations and Special Cases in Flow Aggregation

6.1. Exact versus Approximate Counting during Aggregation

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

6.2. Considerations for Aggregation of Sampled Flows

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

6.3. Considerations for Aggregation of Heterogeneous Flows

Aggregation may be applied to Original Flows from different sources and of different types (i.e., represented using different, perhaps wildly-different Templates). When the goal is to separate the heterogeneous Original Flows and aggregate them into heterogeneous Aggregated Flows, each aggregation should be done at its own Intermediate Aggregation Process. The Observation Domain ID on the Messages containing the output Aggregated Flows can be used to identify the different Processes, and to segregate the output.

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

7. Export of Aggregated IP Flows using IPFIX

In general, Aggregated Flows are exported in IPFIX as any normal

Flow. However, certain aspects of Aggregated Flow export benefit from additional guidelines, or new Information Elements to represent aggregation metadata or information generated during aggregation. These are detailed in the following subsections.

7.1. Time Interval Export

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

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

7.2. Flow Count Export

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

7.2.1. originalFlowsPresent

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

Abstract Data Type: unsigned64

ElementId: TBD1

Status: Current

7.2.2. originalFlowsInitiated

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

Abstract Data Type: unsigned64

ElementId: TBD2

Status: Current

7.2.3. originalFlowsCompleted

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

Abstract Data Type: unsigned64

ElementId: TBD3

Status: Current

7.2.4. deltaFlowCount

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

Abstract Data Type: float64

ElementId: 3

Status: Current

7.3. Distinct Host Export

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

7.3.1. distinctCountOfSourceIPAddress

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

Abstract Data Type: unsigned64

ElementId: TBD4

Status: Current

7.3.2. distinctCountOfDestinationIPAddress

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

Abstract Data Type: unsigned64

ElementId: TBD5

Status: Current

7.3.3. distinctCountOfSourceIPv4Address

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

Abstract Data Type: unsigned32

ElementId: TBD6

Status: Current

7.3.4. distinctCountOfDestinationIPv4Address

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

Abstract Data Type: unsigned32

ElementId: TBD7

Status: Current

7.3.5. distinctCountOfSourceIPv6Address

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

Abstract Data Type: unsigned64

ElementId: TBD8

Status: Current

7.3.6. distinctCountOfDestinationIPv6Address

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

Abstract Data Type: unsigned64

ElementId: TBD9

Status: Current

7.4. Aggregate Counter Distribution Export

When exporting counters distributed among Aggregated Flows, as described in Section 5.1.1, the Exporting Process MAY export an Aggregate Counter Distribution Record for each Template describing Aggregated Flow records; this Options Template is described below. It uses the valueDistributionMethod Information Element, also defined below. Since in many cases distribution is simple, accounting the counters from Contributing Flows to the first Interval to which they contribute, this is default situation, for which no Aggregate Counter Distribution Record is necessary; Aggregate Counter Distribution Records are only applicable in more exotic situations, such as using an Aggregation Interval smaller than the durations of Original Flows.

7.4.1. Aggregate Counter Distribution Options Template

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

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

7.4.2. valueDistributionMethod Information Element

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

Value	Description
1	Start Interval: The counters for an Original Flow are added to the counters of the appropriate Aggregated Flow containing the start time of the Original Flow. This should be assumed the default if value distribution information is not available at a Collecting Process for an Aggregated Flow.
2	End Interval: The counters for an Original Flow are added to the counters of the appropriate Aggregated Flow containing the end time of the Original Flow.
3	Mid Interval: The counters for an Original Flow are added to the counters of a single appropriate Aggregated Flow containing some timestamp between start and end time of the Original Flow.
4	Simple Uniform Distribution: Each counter for an Original Flow is divided by the number of time intervals the Original Flow covers (i.e., of appropriate Aggregated Flows sharing the same Flow Key), and this number is added to each corresponding counter in each Aggregated Flow.

5	Proportional Uniform Distribution: Each counter for an Original Flow is divided by the number of time _units_ the Original Flow covers, to derive a mean count rate. This mean count rate is then multiplied by the number of time units in the intersection of the duration of the Original Flow and the time interval of each Aggregated Flow. This is like simple uniform distribution, but accounts for the fractional portions of a time interval covered by an Original Flow in the first and last time interval.
6	Simulated Process: Each counter of the Original Flow is distributed among the intervals of the Aggregated Flows according to some function the Aggregation Process uses based upon properties of Flows presumed to be like the Original Flow. This is essentially an assertion that the Aggregation Process has no direct packet timing information but is nevertheless not using one of the other simpler distribution methods. The Aggregation Process specifically makes no assertion as to the correctness of the simulation.
7	Direct: The Aggregation Process has access to the original packet timings from the packets making up the Original Flow, and uses these to distribute or recalculate the counters.

Abstract Data Type: unsigned8

ElementId: TBD10

Status: Current

8. Examples

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

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

Figure 8: Input template for examples

The data records given as input to the examples in this section are shown below, in the format "flowStartMilliseconds-flowEndMilliseconds sourceIPv4Address:sourceTransportPort -> destinationIPv4Address: destinationTransportPort (protocolIdentifier) octetDeltaCount"; timestamps are given in H:MM:SS.sss format.

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

Figure 9: Input data for examples

8.1. Traffic Time-Series per Source

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

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

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

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

Figure 10: Output template for time series per source

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

```
                    Original Flows
                          |
                          V
              +-----------------------+
              | interval distribution |
              |  * impose uniform     |
              |    300s time interval |
              +-----------------------+
                  |
                  | Partially Aggregated Flows
                  V
   +------------------------+
   |  key aggregation       |
   |   * reduce key to only |
   |     sourceIPv4Address  |
   +------------------------+
                  |
                  | Partially Aggregated Flows
                  V
             +-------------------------+
             |  aggregate combination  |
             |   * sum octetDeltaCount |
             +-------------------------+
                          |
                          V
                  Aggregated Flows
```

Figure 11: Aggregation operations for time series per source

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

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

Figure 12: Interval imposition for time series per source

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

```
9:00:00.000-9:05:00.000 192.0.2.2      119
9:00:00.000-9:05:00.000 192.0.2.2       83
9:00:00.000-9:05:00.000 192.0.2.2     1637
9:00:00.000-9:05:00.000 192.0.2.3      111
9:00:00.000-9:05:00.000 192.0.2.3    16838
9:00:00.000-9:05:00.000 192.0.2.2    11538
9:00:00.000-9:05:00.000 192.0.2.3      119
9:00:00.000-9:05:00.000 192.0.2.3     2973
9:00:00.000-9:05:00.000 192.0.2.4     8350
9:00:00.000-9:05:00.000 203.0.113.3    778
9:00:00.000-9:05:00.000 203.0.113.3    883
9:05:00.000-9:10:00.000 203.0.113.3   1637
9:05:00.000-9:10:00.000 203.0.113.3    561
9:05:00.000-9:10:00.000 192.0.2.2     1899
9:05:00.000-9:10:00.000 192.0.2.3     1284
9:05:00.000-9:10:00.000 203.0.113.3   2670
9:10:00.000-9:15:00.000 192.0.2.4       75
9:10:00.000-9:15:00.000 192.0.2.4     3374
9:10:00.000-9:15:00.000 192.0.2.4      138
9:10:00.000-9:15:00.000 192.0.2.3     2325
9:10:00.000-9:15:00.000 192.0.2.2     2869
9:10:00.000-9:15:00.000 192.0.2.3    18289
```

Figure 13: Key aggregation for time series per source

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

```
  9:00:00.000-9:05:00.000 192.0.2.2      119
  9:00:00.000-9:05:00.000 192.0.2.2       83
  9:00:00.000-9:05:00.000 192.0.2.2     1637
  9:00:00.000-9:05:00.000 192.0.2.2    11538
+ 9:00:00.000-9:05:00.000 192.0.2.2    15420
                                       -----
= 9:00:00.000-9:05:00.000 192.0.2.2    28797

  9:00:00.000-9:05:00.000 192.0.2.3      111
  9:00:00.000-9:05:00.000 192.0.2.3    16838
  9:00:00.000-9:05:00.000 192.0.2.3      119
+ 9:00:00.000-9:05:00.000 192.0.2.3     2973
                                       -----
= 9:00:00.000-9:05:00.000 192.0.2.3    20041
```

Figure 14: Summation during aggregate combination

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

```
9:00:00.000-9:05:00.000 192.0.2.2    28797
9:00:00.000-9:05:00.000 192.0.2.3    20041
9:00:00.000-9:05:00.000 192.0.2.4     8350
9:00:00.000-9:05:00.000 203.0.113.3  12861
9:05:00.000-9:10:00.000 192.0.2.2     1899
9:05:00.000-9:10:00.000 192.0.2.3     1284
9:05:00.000-9:10:00.000 203.0.113.3   4868
9:10:00.000-9:15:00.000 192.0.2.2     2869
9:10:00.000-9:15:00.000 192.0.2.3    20594
9:10:00.000-9:15:00.000 192.0.2.4     3587
```

Figure 15: Aggregated Flows for time series per source

8.2. Core Traffic Matrix

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

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

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

Figure 16: Example Autonomous system number map

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

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

Figure 17: Output template for traffic matrix

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

```
                 Original Flows
                       |
                       V
            +-----------------------+
            | interval distribution |
            |  * impose uniform     |
            |   3600s time interval |
            +-----------------------+
                   |
                   | Partially Aggregated Flows
                   V
   +------------------------+
   |  key aggregation       |
   |  * reduce key to only  |
   |    sourceIPv4Address + |
   |    destIPv4Address     |
   +------------------------+
                   |
                   V
   +------------------------+
   |  key aggregation       |
   |  * replace addresses   |
   |    with ASN from map   |
   +------------------------+
                   |
                   | Partially Aggregated Flows
                   V
           +--------------------------+
           |  aggregate combination  |
           |   * sum octetDeltaCount |
           +--------------------------+
                       |
                       V
               Aggregated Flows
```

Figure 18: Aggregation operations for traffic matrix

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

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

Figure 19: Interval imposition for traffic matrix

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

```
9:00:00.000-10:00:00.000 AS64496 -> AS64497    119
9:00:00.000-10:00:00.000 AS64496 -> AS64497     83
9:00:00.000-10:00:00.000 AS64496 -> AS64498   1637
9:00:00.000-10:00:00.000 AS64496 -> AS64497    111
9:00:00.000-10:00:00.000 AS64496 -> AS64498  16838
9:00:00.000-10:00:00.000 AS64496 -> AS64498  11538
9:00:00.000-10:00:00.000 AS64496 -> AS64497    119
9:00:00.000-10:00:00.000 AS64496 -> AS64498   2973
9:00:00.000-10:00:00.000 AS64496 -> AS64498   8350
9:00:00.000-10:00:00.000 AS64499 -> AS64498    778
9:00:00.000-10:00:00.000 AS64499 -> AS64498    883
9:00:00.000-10:00:00.000 AS64496 -> AS64498  15420
9:00:00.000-10:00:00.000 AS64499 -> AS64498   1637
9:00:00.000-10:00:00.000 AS64499 -> AS64498    561
9:00:00.000-10:00:00.000 AS64496 -> AS64498   1899
9:00:00.000-10:00:00.000 AS64496 -> AS64498   1284
9:00:00.000-10:00:00.000 AS64499 -> AS64498   2670
9:00:00.000-10:00:00.000 AS64496 -> AS64497     75
9:00:00.000-10:00:00.000 AS64496 -> AS64498   3374
9:00:00.000-10:00:00.000 AS64496 -> AS64498    138
9:00:00.000-10:00:00.000 AS64496 -> AS64498   2325
9:00:00.000-10:00:00.000 AS64499 -> AS64498  11200
9:00:00.000-10:00:00.000 AS64496 -> AS64498   2869
9:00:00.000-10:00:00.000 AS64496 -> AS64498  18289
```

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

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

```
9:00:00.000-10:00:00.000 AS64496 -> AS64497    507
9:00:00.000-10:00:00.000 AS64496 -> AS64498  86934
9:00:00.000-10:00:00.000 AS64499 -> AS64498  17729
```

Figure 21: Aggregated Flows for traffic matrix

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

8.3. Distinct Source Count per Destination Endpoint

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

```
                    Original Flows
                          |
                          V
              +-----------------------+
              | interval distribution |
              |  * discard timestamps |
              +-----------------------+
                  |
                  | Partially Aggregated Flows
                  V
   +-----------------------------+
   |  value aggregation          |
   |  * discard octetDeltaCount |
   +-----------------------------+
                  |
                  | Partially Aggregated Flows
                  V
   +-----------------------------+
   |  key aggregation            |
   |   * reduce key to only      |
   |      destIPv4Address +      |
   |      destTransportPort,     |
   |   * count distinct sources |
   +-----------------------------+
                  |
                  | Partially Aggregated Flows
                  V
       +-----------------------------------------------+
       |  aggregate combination                        |
       |   * no-op (distinct sources already counted) |
       +-----------------------------------------------+
                          |
                          V
                  Aggregated Flows
```

Figure 22: Aggregation operations for source count

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

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

Figure 23: Output template for source count

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

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

```
destination 192.0.2.131:53      3 sources
destination 198.51.100.2:80     1 source
destination 198.51.100.2:443    3 sources
destination 198.51.100.67:80    2 sources
destination 198.51.100.68:80    2 sources
destination 198.51.100.133:80   2 sources
destination 198.51.100.3:80     3 sources
destination 198.51.100.4:80     2 sources
destination 198.51.100.17:80    1 source
destination 198.51.100.69:443   1 source
```

Figure 24: Aggregated flows for source count

8.4. Traffic Time-Series per Source with Counter Distribution

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

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

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

Figure 25: Aggregate Counter Distribution Options Template

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

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

Figure 26: Aggregate Counter Distribution Record

[EDITOR’S NOTE: redo these in boxdiagrams?]

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

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

Figure 27: Distirbuted interval imposition for time series per source

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

```
 * 9:00:00.000-9:05:00.000 192.0.2.2    21087
   9:00:00.000-9:05:00.000 192.0.2.3    20041
   9:00:00.000-9:05:00.000 192.0.2.4     8350
 * 9:00:00.000-9:05:00.000 203.0.113.3   9394
 * 9:05:00.000-9:10:00.000 192.0.2.2     9609
   9:05:00.000-9:10:00.000 192.0.2.3     1284
 * 9:05:00.000-9:10:00.000 203.0.113.3   8601
   9:10:00.000-9:15:00.000 192.0.2.2     2869
   9:10:00.000-9:15:00.000 192.0.2.3    20594
   9:10:00.000-9:15:00.000 192.0.2.4     3587
 * 9:10:00.000-9:15:00.000 203.0.113.3   3734
```

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

9. Security Considerations

This document specifies the operation of an Intermediate Aggregation Process with the IPFIX Protocol; the Security Considerations for the protocol itself in Section 11 of [RFC5101] therefore apply. In the common case that aggregation is performed on a Mediator, the Security Considerations for Mediators in Section 9 of [RFC6183] apply as well.

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

10. IANA Considerations

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

[NOTE for IANA: The text TBDn should be replaced with the respective assigned Information Element numbers where they appear in this document. Note that the deltaFlowCount Information Element has been assigned the number 3, as it is compatible with the corresponding existing (reserved) NetFlow v9 Information Element. Other

Information Element numbers should be assigned outside the NetFlow V9 compatibility range, as these Information Elements are not supported by NetFlow V9.]

11. Acknowledgments

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

12. References

12.1. Normative References

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

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

12.2. Informative References

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

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

[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., Stiemerling, M., and P. Aitken, "IP Flow Information Export (IPFIX) Implementation Guidelines", RFC 5153, April 2008.

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

[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.

[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.

[RFC5835] Morton, A. and S. Van den Berghe, "Framework for Metric Composition", RFC 5835, April 2010.

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

[RFC6183] Kobayashi, A., Claise, B., Muenz, G., and K. Ishibashi, "IP Flow Information Export (IPFIX) Mediation: Framework", RFC 6183, April 2011.

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

[I-D.claise-ipfix-mediation-protocol]
Claise, B., Kobayashi, A., and B. Trammell, "Specification of the Protocol for IPFIX Mediations", draft-claise-ipfix-mediation-protocol-04 (work in progress), July 2011.

[I-D.trammell-ipfix-ie-doctors]
Trammell, B. and B. Claise, "Guidelines for Authors and Reviewers of IPFIX Information Elements", draft-trammell-ipfix-ie-doctors-02 (work in progress), June 2011.

[I-D.ietf-ipfix-configuration-model]
Muenz, G., Claise, B., and P. Aitken, "Configuration Data Model for IPFIX and PSAMP", draft-ietf-ipfix-configuration-model-10 (work in progress), July 2011.

[I-D.ietf-ipfix-flow-selection-tech]
D’Antonio, S., Zseby, T., Henke, C., and L. Peluso, "Flow

Selection Techniques",
draft-ietf-ipfix-flow-selection-tech-07 (work in
progress), July 2011.

Authors' Addresses

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

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

Arno Wagner
Consecom AG
Bleicherweg 64a
8002 Zurich
Switzerland

Email: arno@wagner.name

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

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

IPFIX Working Group B. Trammell
Internet-Draft ETH Zurich
Intended status: BCP B. Claise
Expires: April 30, 2012 Cisco Systems, Inc.
October 28, 2011

Guidelines for Authors and Reviewers of IPFIX Information Elements
draft-trammell-ipfix-ie-doctors-03.txt

Abstract

This document provides guidelines for the definition of IPFIX Information Elements for addition to the IANA IPFIX Information Element registry, in order to extend the applicability of the IPFIX protocol to new operations and management areas.

Status of this Memo

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

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

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

This Internet-Draft will expire on April 30, 2012.

Copyright Notice

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

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

Table of Contents

1. Introduction

This document provides guidelines for the extension of the applicability of the IP Flow Information Export (IPFIX) protocol to network operations and management purposes outside the initial scope defined in "IPFIX Applicability Statement" [RFC5472]. These new applications are largely defined by creating new Information Elements beyond those in the IANA IPFIX Information Element Registry [iana-ipfix-assignments]. New applications may be further specified through additional RFCs defining and describing their usage.

We intend this document to enable the expansion of the applicability of IPFIX to new areas by experts in the working group or area directorate concerned with the technical details of the protocol or application to be measured or managed using IPFIX. This expansion would occur with the consultation of IPFIX experts informally called 'IE-Doctors'. It provides guidelines both for those defining new Information Elements as well as the IE-Doctors reviewing them.

1.1. Intended Audience and Usage

This document is meant for two separate audiences. For IETF contributors extending the applicability of IPFIX, it provides a set of guidelines and best practices to be used in deciding which Information Elements are necessary for a given existing or new application, defining these Information Elements, and deciding whether an RFC should be published to further describe the application. For the IPFIX experts appointed as IE-Doctors, and for IANA personnel changing the Information Element registry, it defines a set of acceptance criteria against which these proposed Information Elements should be evaluated.

This document is not intended to guide the extension of the IPFIX protocol itself, e.g. through new export mechanisms, data types, or the like; these activities should be pursued through the publication of standards-track RFCs by the IPFIX Working Group.

This document specifies additional practices beyond those appearing in the IANA Considerations sections of existing IPFIX documents, especially the Information Model [RFC5102]. The practices outlined in this document are intended to guide experts when making changes to the IANA registry under Expert Review as defined in [RFC5226].

1.2. Overview of relevant IPFIX documents

[RFC5101] defines the IPFIX Protocol, the IPFIX-specific terminology used by this document, and the data type encodings for each of the data types supported by IPFIX.

[RFC5102] defines the initial IPFIX Information Model, as well as procedures for extending the Information Model. It states that new Information Elements may be added to the Information Model on Expert Review basis, and delegates the appointment of experts to an IESG Area Director. This document is intended to further codify the best practices to be followed by these experts, in order to improve the efficiency of this process.

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

[RFC5610] defines a method for exporting information about Information Elements inline within IPFIX. In doing so, it explicitly defines a set of restrictions on the use of data types and semantics which are implied in [RFC5101] and [RFC5102]; 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 [RFC5101] are to be interpreted as defined there.

An "application", as used in this document, refers to a candidate protocol, task, or domain to which IPFIX export, collection, and/or storage is applied, beyond those within the IPFIX Applicability statement [RFC5472]. By this definition, PSAMP [RFC5476] was the first new IPFIX application after the publication of the IPFIX protocol [RFC5101].

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

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

3. How to apply IPFIX

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

- o The application's data flow is fundamentally unidirectional. IPFIX is a "push" protocol, supporting only the export of information from a sender (an Exporting Process) to a receiver (a Collecting Process). Request-response interactions are not supported by IPFIX.

- o The application handles discrete event information, or information to be periodically reported. IPFIX is particularly well suited to representing events, which can be scoped in time.

- o The application handles information about network entities. IPFIX's information model is network-oriented, so network management applications have many opportunities for information model reuse.

- o The application requires a small number of arrangements of data structures relative to the number of records it handles. The template-driven self-description mechanism used by IPFIX excels at handling large volumes of identically structured data, compared to representations which define structure inline with data (such as XML).

Most applications meeting these criteria can be supported over IPFIX. Once it's been determined that IPFIX is a good fit, the next step is determining which Information Elements are necessary to represent the information required by the application. Especially for network-centric applications, the IPFIX Information Element registry may already contain all the necessary Information Elements (see Section 6.1 for guidelines on maximizing Information Element reuse). In this case, no additional work within the IETF is necessary: simply define Templates and start exporting.

It is expected, however, that most applications will be able to reuse some existing Information Elements, but must define some additional Information Elements to support all their requirements; in this case, see Section 4 for best practices to be followed in defining Information Elements.

Optionally, a Working Group or individual contributor may choose to publish an RFC detailing the new IPFIX application. Such an RFC should contain discussion of the new application, the Information

Element definitions as in Section 4, as well as suggested Templates and examples of the use of those Templates within the new application as in Section 8.2. Section 9 defines a compact textual Information Element notation to be used in describing these suggested Templates and/or the use of IPFIX Structured Data [I-D.ietf-ipfix-structured-data] within the new application.

4. Defining new Information Elements

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

- o The Information Element MUST be sufficiently unique within the registry. A proposed Information Elements which is a substantial duplicate of an exiting Information Element is to be represented using the existing Element.

- o The Information Element SHOULD contain minimal internal structure; complex information should be represented with multiple simple Information Elements to be exported in parallel, as in Section 4.5.

- o The Information Element SHOULD be generally applicable to the application at hand, which SHOULD be of general interest to the community. Information Elements representing information about proprietary or nonstandard applications SHOULD be represented using enterprise-specific Information Elements as detailed in section 6.2 of [RFC5101].

The definition of new Information Elements requires a descriptive name, a specification of the data type as one from the IPFIX Data Type Registry, and a human-readable description written in English. This section provides guidelines on each of these components of an Information Element definition, referring to existing documentation such as [RFC5102] as appropriate.

4.1. Information Element naming

Information Element Names should be defined in accordance with section 2.3 of [RFC5102]; the most important naming conventions are repeated here for convenience.

- Names of Information Elements should be descriptive.

- Names of Information Elements MUST be unique within the IPFIX information model.

- Names of Information Elements start with non-capitalized letters.

- Composed names use capital letters for the first letter of each component (except for the first one). All other letters are non-capitalized, even for acronyms. Exceptions are made for acronyms containing non-capitalized letter, such as 'IPv4' and 'IPv6'. Examples are sourceMacAddress and destinationIPv4Address.

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

4.2. Information Element data types

IPFIX provides a set of data types covering most primitives used in network measurement and management applications. The most appropriate data type should be chosen for the Information Element type, out of the IPFIX informationElementDataTypes subregistry at [iana-ipfix-assignments].

Because IPFIX provides reduced-length encoding for Information Elements, unless an integral Information Element is derived from a fixed-width field in a measured protocol (e.g., tcpSequenceNumber, which is an unsigned32), it should be defined with the maximum possible width, generally signed64 or unsigned64. Applications can then choose to use reduced-size encoding as defined in Section 6.2 of [RFC5101] in cases where fewer than 2^64 values are necessary.

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

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 (i.e., an IPFIX application that exports Options Template Records mapping IP addresses to additional information about each host from an external database MUST use Information Elements of an address type to represent the addresses, even if the source database represented these as strings.)

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

4.3. Information Element numbering

In general, when adding newly registered Information Elements to the registry, IANA SHOULD assign the lowest available Information Element identifier (the value column in [iana-ipfix-assignments] in the range 128-32767, noting that prior noncontiguous allocation may lead to unassigned Information Elements with lower Information Element identifiers than some presently assigned Information Elements. This is the case with the PSAMP Information Model [RFC5477], which assigned a block of Information Elements identifiers starting at 300.

Information Element identifiers in the range 1-128 MUST NOT be assigned unless the Information Element is compatible with the NetFlow v9 protocol as described in [RFC3954]. Such Information Elements may ONLY be requested by a NetFlow v9 expert, to be designated by the IESG to consult with IANA on NetFlow v9 compatibility with IPFIX.

4.4. Ancillary Information Element properties

Information Elements to which special semantics apply SHOULD define these semantics with one of the values in the Information Element Semantics registry, as described in Section 3.2 of [RFC5102], subject to the restrictions given in Section 3.10 of [RFC5610]; essentially, the semantics and the type must be consistent.

When defining Information Elements representing a dimensioned quantity or entity count, the units of that quantity SHOULD be defined in the units field. This field takes its values from the IANA Information Element Units registry. If an Information Element expresses a quantity in units not yet in this registry, then the unit must be added to the Units registry at the same time the Information Element is added to the Information Element registry.

Additionally, when the range of values an Information Element can take is smaller than the range implied by its data type, the range SHOULD be defined within the Information Element registry.

4.5. Internal structure in Information Elements

The definition of Information Elements with internal structure with the structure defined in the Description field is discouraged, except in the following cases:

- 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)

- 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)

- The Information Element represents a set of flags which are tightly semantically related, where representing the flags as separate one-byte booleans would be inefficient, and which should always appear together in a data record (e.g., the anonymizationFlags Information Element for specifying optional features of anonymization techniques)

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

As an example of information element decomposition, consider an application-level identifier called an "endpoint", which represents a {host, port, protocol} tuple. Instead of allocating an opaque, structured "source endpoint" Information Element, the source endpoint should be represented by three separate Information Elements: "source address", "source port", "transport protocol". In this example, the required information elements already exist in the Information Element registry: sourceIPv4Address or sourceIPv6Address, sourceTransportPort, protocolIdentifier. Indeed, as well as being good practice, this normalization down to non-structured Information Elements also increases opportunities for reuse as in Section 6.1.

The decomposition of data with internal structure SHOULD avoid the definition of Information Elements with a meaning too specific to be generally useful, or that would result in either the export of meaningless data or a multitude of templates to handle different multiplicities. More information on multiplicities is given in the following section.

4.6. Information Element multiplicity

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

- o As specified in section 8 of [RFC5101]: "if an Information Element is required more than once in a Template, the different occurrences of this Information Element SHOULD follow the logical order of their treatments by the Metering Process." In other words, in cases where the items have a natural order (e.g., the order in which they occur in the packet), and the multiplicity is the same for each record, the information can be modeled by containing multiple instances of the Information Element representing a single item within the Template Record describing the Data Records.

- o In cases where the items have a variable multiplicity, a basicList of the Information Element representing a single item can be used as in the IPFIX Structured Data [I-D.ietf-ipfix-structured-data] extension.

- o If the multiple-item structure is taken directly from bytes observed on the wire by the Metering Process or otherwise taken from the application being measured, the multiple-item structure can be exported as a variable-length octetArray Information Element holding the raw content.

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

4.7. Enumerated Values and Subregistries

When defining an Information Element that takes an enumerated value from a set of values which may change in the future, this enumeration MUST be defined by an IANA registry or subregistry. For situations where an existing registry defines the enumeration (e.g., the IANA Protocol Numbers registry for the protocolIdentifier Information Element), that registry MUST be used. Otherwise, a new IPFIX subregistry must be defined for the enumerated value, to be modified subject to Expert Review [RFC5226].

4.8. Reversibility as per RFC 5103

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

New non-reversible Information Elements SHOULD contain a note in the description stating that they are not reversible.

4.9. Promotion of Enterprise-Specific Information Elements

Some Information Elements may start their lifecycle outside the IANA registry as enterprise-specific Information Elements scoped to a Private Enterprise Number. One stated goal of enterprise-specific Information Elements is pre-standards product delivery and experimentation; should these experiments be successful and the Information Elements generally useful, these SHOULD subsequently registered with IANA.

In order to support transition from experimental registration to IANA registration, the IANA registry provides an optional "enterprise-specific IE reference" column for each Information Element. In cases of promoted enterprise-specific Information Elements, this column in the registry SHOULD contain the private enterprise and Information Element numbers of the enterprise-specific version of the Information Element.

4.10. Avoiding Bad Ideas in Information Element Design

In general, the existence of a similarly-defined Information Element in the IANA registry sets a precedent which may be followed to determine whether a given proposed Information Element "fits" within the registry. Indeed, the rules specified by this document could be interpreted to mean "make new Information Elements that look like existing Information Elements". However, for reasons of history, there are several Information Elements within the IANA registry which do not follow best practices in Information Element design, and should be explicitly ignored when looking for guidance as to whether a new Information Element should be added.

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

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

As mentioned in Section 4.6, an Information Element definition SHOULD NOT include any ordinality or multiplicity information. The only example of this within the IANA registry the following list of assigned IPFIX Information Elements: mplsTopLabelStackSection, mplsLabelStackSection2, mplsLabelStackSection3, mplsLabelStackSection4, mplsLabelStackSection5, mplsLabelStackSection6 mplsLabelStackSection7, mplsLabelStackSection8, mplsLabelStackSection9, and mplsLabelStackSection10. The only distinction between those almost-identical Information Elements is the position within the MPLS stack. This Information Element design pattern met an early requirement of the definition of IPFIX which was not carried forward into the final specification -- namely, that no semantic dependency was allowed between Information Elements in the same Record -- and as such SHOULD NOT be followed in the definition of new Information Elements. In this case, since the size of the MPLS stack will vary from flow to flow, it should be exported using IPFIX Structured Data [I-D.ietf-ipfix-structured-data] where supported, as a basicList of MPLS label entries, or as a raw MPLS label stack using the variable-length mplsLabelStackSection Information Element.

5. The Information Element Lifecycle

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

5.1. The IE-DOCTORS process

Requests to change the IANA Information Element registry or a linked subregistry are submitted to IANA, which forwards the request to a designated group of experts (IE-DOCTORS) appointed by the IETF Operations Area Directors. This group of experts reviews the request for compliance with this document, compliance with other applicable IPFIX-related RFCs, and consistency with the currently defined set of Information Elements.

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

5.2. Revising Information Elements

The Information Element status field in the Information Element Registry is defined in [RFC5102] to allow Information Elements to be 'current', 'deprecated' or 'obsolete'. No Information Elements are as of this writing deprecated or obsolete, and [RFC5102] does not define any policy for using them. Additionally, no policy is defined for revising Information Element registry entries or addressing errors therein. To be certain, changes and deprecations within the Information Element registry are not encouraged, and should be avoided to the extent possible. However, in recognition that change is inevitable, this section is intended to remedy this situation.

The primary requirement in the definition of a policy for managing changes to existing Information Elements is avoidance of interoperability problems; IPFIX experts appointed to review changes to the Information Element Registry MUST work to maintain interoperability above all else. Changes to Information Elements already in use may only be done in an interoperable way; necessary changes which cannot be done in a way to allow interoperability with unchanged implementations MUST result in deprecation.

A change to an Information Element is held to be interoperable only when:

o it involves the correction of an error which is obviously only editorial; or

- it corrects an ambiguity in the Information Element's definition, which itself leads to non-interoperability (e.g., a prior change to ipv6ExtensionHeaders); or

- it expands the Information Element's data type without changing how it is represented (e.g., changing unsigned32 to unsigned64, as with a prior change to selectorId); or

- it defines a previously undefined or reserved enumerated value, or one or more previously reserved bits in an Information Element with flag semantics; or

- it expands the set of permissible values in the Information Element's range; or

- it harmonizes with an external reference which was itself corrected.

A non-interoperable Information Element change may also be made if it can be reasonably assumed in the eyes of the appointed experts that no unchanged implementation of the Information Element exists; this can be held to happen if a non-interoperable change to an Information Element defined shortly before is proposed to the IPFIX mailing list by the original proposer of the Information Element, and no objection is raised within a reasonable amount of time, to be defined by the expert reviewers.

If a change is permissible, it is sent to IANA, which passes it to the appointed experts for review; if there is no objection to the change from any appointed expert, IANA makes the change in the Information Element Registry. The requestor of the change is appended to the Requestor in the registry.

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

5.3. Deprecating Information Elements

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

- the Information Element definition has an error or shortcoming which cannot be permissibly changed as above; or

- the deprecation harmonizes with an external reference which was itself deprecated through that reference's accepted deprecation method; or

- changes in the IPFIX Protocol or its extensions, or in community understanding thereof, allow the information represented by the Information Element to be represented in a more efficient or convenient way. Deprecation in this circumstance additionally requires the assent of the IPFIX Working Group, and should be specified in the Internet Draft(s) defining the protocol change.

A request for deprecation is sent to IANA, which passes it to the IE-DOCTORS for review, as above. When deprecating an Information Element, the Information Element description MUST be updated to explain the deprecation, as well as to refer to any new Information Elements created to replace the deprecated Information Element. The revision number of an Information Element is incremented upon deprecation.

Deprecated Information Elements SHOULD continue to be supported by Collecting Processes, but SHOULD NOT be exported by Exporting Processes. The use of deprecated Information Elements SHOULD result in a log entry or human-readable warning at the Exporting and Collecting Processes. After a period of time determined in the eyes of the IE-DOCTORS experts to be reasonable in order to allow deployed Exporting Processes to be updated to account for the deprecation, a deprecated Information Element may be made obsolete. Obsolete Information Elements MUST NOT be supported by either Exporting or Collecting Processes. The receipt of obsolete Information Elements SHOULD be logged by the Collecting Process.

Names of deprecated Information Elements MUST NOT be reused. Names of obsolete Information Elements MAY be reused, but this is NOT RECOMMENDED, as it may cause confusion among users.

5.4. Versioning the entire IANA Registry

Consider a typical Collector implementation, which regularly downloads the entire registry in order to be compliant with the latest of set of supported IEs. While a registry revision number might seems advantageous for the Collector at first glance (avoiding the one by one comparison of all IE revisions), it is not necessary, as the IPFIX IANA registry specifies the date at which the registry was last updated in the "Last Updated" field. For purposes of identifying the latest set of Information Element versions specified in registry, the last revision date of the Information Element registry (available in the registry XML source, or from the Last-Modified: header of [iana-ipfix-assignments]) SHOULD be used.

6. When not to define new Information Elements

Also important in defining new applications is avoiding redundancy and clutter in the Information Element registry. Here we provide guidelines for reuse of existing Information Elements, as well as guidelines on using enterprise-specific Information Elements instead of adding Information Elements in the registry.

6.1. Maximizing reuse of existing Information Elements

Whenever possible, new applications should prefer usage of existing IPFIX Information Elements to the creation of new Information Elements. IPFIX already provides Information Elements for every common Layer 4 and Layer 3 packet header field in the IETF protocol suite, basic Layer 2 information, basic counters, timestamps and time ranges, and so on. When defining a new Information Element similar to an existing one, reviewers shall ensure that the existing one is not applicable.

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

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

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

Specifically, 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 Infomation Elements, as in Section 3.2 of [RFC5101]. These are scoped to a vendor's or organization's Structure of Management Information (SMI) Private Enterprise Number, and are under complete control of the organization assigning them.

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

- o export of implementation-specific information, or
- o export of information derived in a commercially-sensitive or proprietary method, or
- o export of information or meta-information specific to a commercially-sensitive or proprietary application.

While work within the IETF generally does not fall into these categories, enterprise-specific Information Elements are also useful for pre-standardization testing of a new IPFIX application. While performing initial development and interoperability testing of a new application, the Information Elements used by the application SHOULD NOT be submitted to IANA for inclusion in the registry. Instead, these experimental Information Elements SHOULD be represented as enterprise-specific until their definitions are finalized, then transitioned from enterprise-specific to IANA-defined upon finalization. To support this transition, the IANA registry provides an experimental IE reference as defined in Section 4.9.

7. Applying IPFIX to non-Flow Applications

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

Non-flow information defining associations or key-value pairs, on the other hand, are 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 Informatin 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.

8. Writing Internet-Drafts for IPFIX Applications

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

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

When defining new Information Elements in an Internet-Draft, the Internet-Draft SHOULD contain a section (or subsection) for each Information Element, which contains the attributes in Section 4 in human-readable form. An example subsection is given below. These Information Element descriptions SHOULD NOT assign Information Element numbers, instead using placeholder identifiers for these numbers (e.g. "AAA", "BBB", "CCC", or "TBD1", "TBD2", "TBD3") and a note to IANA in the IANA Considerations section to replace those placeholders in the document with Information Element numbers when the numbers are assigned. The use of these placeholder definitions allows references to the numbers in e.g. box-and-line diagrams or template definitions as in Section 9.

8.1. Example Information Element Definition

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

Description: Description goes here.

Data Type: Data type goes here; obligatory

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

Units: Units, if any, go here; optional

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

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

ElementId: TBD1

8.2. Defining Recommended Templates

New IPFIX applications SHOULD NOT, in the general case, define fixed templates for export, as this throws away much of the flexibility afforded by IPFIX. However, fixed template export is permissible in the case that the export implementation must operate in a resource constrained environment, and/or that the application is replacing an existing fixed-format binary export format in a maximally compatible way. In any case, Collecting Processes for such applications SHOULD support reordered Templates or Templates with additional Information Elements.

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

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

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

Recommended Templates are defined, for example, in [RFC5476] for PSAMP packet reports (section 6.4) and extended packet reports (section 6.5). Recommended Options Templates are defined extensively throughout the IPFIX documents, including in the protocol document itself [RFC5101] for exporting export statistics; in the file format [RFC5655] for exporting file metadata; and in Mediator intermediate process definitions such as [I-D.ietf-ipfix-anon] for intermediate process metadata. The discussion in these examples is a good model for recommended template definitions.

9. A Textual Format for Specifying Information Elements and Templates

The examples given above are all expressed using bitmap diagrams of the respective Templates. These are illustrative of the wire representation of simple Templates, but not particularly readable for more complicated recommended Templates, provide no support for rapid implementation of new Templates, and do not adequately convey the optional nature of ordering and additional Information Elements as above. Therefore, we define a RECOMMENDED textual format for specifying Information Elements and Templates in Internet-Drafts in this section.

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

9.1. Information Element Specifiers

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

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

- () parentheses for Information Element numbers,
- <> angles for Information Element data types, and
- [] square brackets for Information Element sizes.
- {} curly braces contain an optional space-separated list of context identifiers to be associated with an Information Element, as described in more detail in Section 9.2

The symbol + is reserved for Information Element nesting within structured data elements; these are described in and Section 9.3, respectively.

Whitespace in IESpecs is insignificant; spaces can be added after each element in order, e.g., to align columns for better readability.

The basic form of a fully-qualified IESpec for an IANA-registered Information Element is as follows:

name(number)<type>[size]

where 'name' is the name of the Information Element in UTF-8, 'number' is the Information Element as a decimal integer, 'type' is the name of the data type as in the IANA informationElementDataTypes registry, and 'size' is the length of the Information Element in octets as a decimal integer, where 65535 or the string 'v' signifies a variable-length Information Element. [size] may be omitted; in this case, the data type's native or default size is assumed.

The basic form of a fully-qualified IESpec for an enterprise-specific Information Element is as follows:

name(pen/number)<type>[size]

where 'pen' is the Private Enterprise Number as a decimal integer.

A fully-qualified IESpec is intended to express enough information about an Information Element to decode and display Data Records defined by Templates containing that Information Element. Range, unit, semantic, and description information, as in [RFC5610], is not supported by this syntax.

Example fully-qualified IESpecs follow:

octetDeltaCount(1)<unsigned64>[8]

octetDeltaCount(1)<unsigned64> (unsigned64 is natively 8 octets long)

sourceIPv4Address(8)<ipv4Address>

wlanSSID(146)<string>[v]

sipRequestURI(35566/403)<string>[65535]

A partial IESpec is any IESpec that is not fully-qualified; these are useful when defining templates. A partial IESpec is assumed to take missing values from its canonical definition, for example, the IANA registry. At minimum, a partial IESpec must contain a name, or a number. Any name, number, or type information given with a partial IESpec must match the values given in the Information Model; however, size information in a partial IESpec overrides size information in the Information Model; in this way, IESpecs can be used to express reduced-length encoding for Information Elements.

Example partial IESpecs follow:

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)

9.2. Specifying Templates

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

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

An Options Template is specified similarly. Scope is specified appending the {scope} context to the end of each IESpec for a Scope IE. Due to the way Information Elements are represented in Options Templates, all {scope} IESpecs must appear before any non-scope IESpec. The Flow Key Options Template defined in section 4.4 of [RFC5101] in IESpec syntax is shown below:

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

9.3. Specifying IPFIX Structured Data

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

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

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

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

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

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

10. Security Considerations

The security aspects of new Information Elements must be considered in order not to give a potential attacker too much information. For example, the "A Framework for Packet Selection and Reporting" [RFC5474] concluded in section 12.3.2 that the hash functions private parameters should not exported within IPFIX.

If some security considerations are specific to an Information Element, they MUST be mentioned in the Information Element description. For example, the ipHeaderPacketSection in the IPFIX registry mentions: "This Information Element, which may have a variable length, carries a series of octets from the start of the IP header of a sampled packet. With sufficient length, this element also reports octets from the IP payload, subject to [RFC2804]. See the Security Considerations section."

These security considerations MAY also be stressed in an accompanying Internet-Draft, as in Section 8. For example, the "Packet Sampling (PSAMP) Protocols Specification" [RFC5476] specifies: "In the basic

Packet Report, a PSAMP Device exports some number of contiguous bytes from the start of the packet, including the packet header (which includes link layer, network layer and other encapsulation headers) and some subsequent bytes of the packet payload. The PSAMP Device SHOULD NOT export the full payload of conversations, as this would mean wiretapping [RFC2804]. The PSAMP Device MUST respect local privacy laws."

11. IANA Considerations

With respect to the management of the IPFIX Information Element registry and associated subregistries located at [iana-ipfix-assignments], this document defines a process for IANA in Section 5.1, and includes a set of guidelines for IANA for applying this process in Section 4, Section 5, and Section 6.

In addition, in order to support more effective management of the Information Element lifecycle as defined in Section 5, it specifies the addition of three new columns for this registry:

Revision: a serial revision number for each Information Element, beginning at 0 for all presently existing and newly created Information Elements.

Date: the date at which the Information Element was created or last modified.

Enterprise-specific reference: for Information Elements which where deployed as enterprise-specific Information Elements for experimentation and testing, and subsequently registered in the IANA registry, specifies the private enterprise number (PEN) and IE number of the equivalent experimental IE.

12. Acknowledgements

The authors would like to acknowledge the FP7 PRISM and DEMONS projects for their material support of this work.

13. References

13.1. Normative References

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

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

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

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

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

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

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

13.2. Informative References

[RFC2804] IAB and IESG, "IETF Policy on Wiretapping", RFC 2804, May 2000.

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

[RFC4181] Heard, C., "Guidelines for Authors and Reviewers of MIB Documents", BCP 111, RFC 4181, September 2005.

[RFC5153] Boschi, E., Mark, L., Quittek, J., Stiemerling, M., and P. Aitken, "IP Flow Information Export (IPFIX) Implementation Guidelines", RFC 5153, April 2008.

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

[RFC5471] Schmoll, C., Aitken, P., and B. Claise, "Guidelines for IP Flow Information Export (IPFIX) Testing", RFC 5471, March 2009.

[RFC5472] Zseby, T., Boschi, E., Brownlee, N., and B. Claise, "IP

Flow Information Export (IPFIX) Applicability", RFC 5472, March 2009.

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

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

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

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

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

[I-D.ietf-ipfix-structured-data]
Claise, B., Dhandapani, G., Yates, S., and P. Aitken, "Export of Structured Data in IPFIX", draft-ietf-ipfix-structured-data-06 (work in progress), May 2011.

[I-D.ietf-ipfix-anon]
Boschi, E. and B. Trammell, "IP Flow Anonymization Support", draft-ietf-ipfix-anon-06 (work in progress), January 2011.

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

Authors' Addresses

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

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

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

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

Network Working Group
Internet-Draft
Intended status: Informational
Expires: May 3, 2012
A. Yourtchenko
P. Aitken
B. Claise
Cisco Systems, Inc.
October 31, 2011

Cisco Specific Information Elements for IPFIX
draft-yourtchenko-cisco-ies-02

Abstract

This document describes some additional Information Elements of Cisco Systems, Inc. that are not listed in RFC3954.

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 3, 2012.

Copyright Notice

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

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

Table of Contents

1. Introduction

The section 4 of [RFC5102] defines the IPFIX Information Elements in the range of 1-127 to be compatible with the NetFlow version 9 fields, as specified in the "Cisco Systems NetFlow Services Export Version 9" [RFC3954]. As [RFC3954] was specified in 2004, it does not contain all NetFlow version 9 specific fields in the range 1-127. The question was asked whether IPFIX Devices should exclusively report the IPFIX IANA IEs [IPFIX-IANA] ? In other words, when upgrading from a NetFlow metering process to an IPFIX Metering Process, should the IPFIX Devices stop reporting NetFlow version 9 specific IEs that were not registered in IANA [IPFIX-IANA] ?

This document is intended to fill the gap in this IE range. That way, IPFIX implementations could export all the IEs specified in IANA, regardless of the range.

2. Terminology

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

3. Information Elements Overview

The following Information Elements are discussed in the sections below:

ID	Name	ID	Name
3	deltaFlowCount	84	samplerName
34	samplingInterval	87	flagsAndSamplerId
35	samplingAlgorithm	89	forwardingStatus
38	engineType	92	srcTrafficIndex
39	engineId	93	dstTrafficIndex
43	ipv4RouterSc	100	className
48	samplerId	102	layer2packetSectionOffset
49	samplerMode	103	layer2packetSectionSize
50	samplerRandomInterval	104	layer2packetSectionData
51	classId		

Table 1

4. Information Elements

4.1. deltaFlowCount

Description:
This Information Element specifies the current number of all Flow Records that form the parent population as input to the Flow Selection Process.
Abstract Data Type: unsigned64
ElementId: 3
Semantics: quantity
Status: current
Units: flows
RFC EDITOR NOTE: if the Flow Aggregation for IPFIX document [I-D.trammell-ipfix-a9n] is published before this, then remove this entry. This Information Element is similar to 'deltaFlowCount' there.

4.2. samplingInterval

Description:
Deprecated in favor of 305 samplingPacketInterval. When using sampled NetFlow, the rate at which packets are sampled - e.g. a value of 100 indicates that one of every 100 packets is sampled.
Abstract Data Type: unsigned32
ElementId: 34
Semantics: quantity
Status: deprecated
Units: packets

4.3. samplingAlgorithm

Description:
Deprecated in favor of 304 selectorAlgorithm. The type of algorithm used for sampled NetFlow:
1 - Deterministic Sampling;
2 - Random Sampling.
The values are not compatible with the selectorAlgorithm IE, where "Deterministic" has been replaced by "Systematic count-based" (1) or "Systematic time-based" (2), and "Random" is (3). Conversion is required, see PSAMP parameters [PSAMP-IANA].
Abstract Data Type: unsigned8
ElementId: 35
Semantics: identifier

Status: deprecated

4.4. engineType

Description:
Type of flow switching engine in a router/switch:
RP = 0,
VIP/Line card = 1,
PFC/DFC = 2.
Reserved for internal use on the collector.
Abstract Data Type: unsigned8
ElementId: 38
Semantics: identifier
Status: deprecated

4.5. engineId

Description:
VIP or line card slot number of the flow switching engine in a router/switch. Reserved for internal use on the collector.
Abstract Data Type: unsigned8
ElementId: 39
Semantics: identifier
Status: deprecated

4.6. ipv4RouterSc

Description:
This is a platform-specific field for Catalyst 5000/Catalyst 6000 family. It is used to store the address of a router that is being shortcut when performing MultiLayer Switching.
Abstract Data Type: ipv4Address
ElementId: 43
Semantics: ipv4Address
Status: deprecated
Reference: [CCO-MLS] describes the MultiLayer Switching.

4.7. samplerId

Description:
Deprecated in favor of 302 selectorId. The unique identifier associated with samplerName.
Abstract Data Type: unsigned8
ElementId: 48

Semantics: identifier
Status: deprecated

4.8. samplerMode

Description:
Deprecated in favor of 304 selectorAlgorithm. The values are not compatible: selectorAlgorithm=3 is random sampling. The type of algorithm used for sampling data: 1 - deterministic, 2 - random sampling. Use with samplerRandomInterval.
Abstract Data Type: unsigned8
ElementId: 49
Semantics: identifier
Status: deprecated

4.9. samplerRandomInterval

Description:
Deprecated in favour of 305 samplingPacketInterval. Packet interval at which to sample - in case of random sampling. Used in connection with samplerMode 0x02 (random sampling) value.
Abstract Data Type: unsigned32
ElementId: 50
Semantics: quantity
Status: deprecated

4.10. classId

Description:
Deprecated in favour of 302 selectorId. Characterizes the traffic class, i.e. QoS treatment.
Abstract Data Type: unsigned8
ElementId: 51
Semantics: identifier
Status: deprecated

4.11. samplerName

Description:
Deprecated in favor of 335 selectorName. Name of the flow sampler.
Abstract Data Type: string
ElementId: 84
Status: deprecated

4.12. flagsAndSamplerId

Description:
Flow flags and the value of the sampler ID (samplerId) combined in one bitmapped field. Reserved for internal use on the collector.
Abstract Data Type: unsigned32
ElementId: 87
Semantics: identifier
Status: deprecated

4.13. forwardingStatus

Description:
This Information Element describes the forwarding status of the flow and any attached reasons. The Reduced Size Encoding rules as per [RFC5101] apply.

The basic encoding is 8 bits. The future extensions could add one or three bytes. The layout of the basic encoding is as follows:

```
MSB -   0   1   2   3   4   5   6   7   - LSB
      +---+---+---+---+---+---+---+---+
      | Status|  Reason code or flags |
      +---+---+---+---+---+---+---+---+
```

Status:

00b = Unknown
01b = Forwarded
10b = Dropped
11b = Consumed

Reason Code (status = 01b, Forwarded)

01 000000b = 64 = Unknown
01 000001b = 65 = Fragmented
01 000010b = 66 = Not Fragmented

Reason Code (status = 10b, Dropped)

10 000000b = 128 = Unknown
10 000001b = 129 = ACL deny
10 000010b = 130 = ACL drop
10 000011b = 131 = Unroutable
10 000100b = 132 = Adjacency
10 000101b = 133 = Fragmentation and DF set

```
   10 000110b = 134 = Bad header checksum
   10 000111b = 135 = Bad total Length
   10 001000b = 136 = Bad header length
   10 001001b = 137 = bad TTL
   10 001010b = 138 = Policer
   10 001011b = 139 = WRED
   10 001100b = 140 = RPF
   10 001101b = 141 = For us
   10 001110b = 142 = Bad output interface
   10 001111b = 143 = Hardware

   Reason Code (status = 11b, Consumed)

   11 000000b = 192 = Unknown
   11 000001b = 193 = Punt Adjacency
   11 000010b = 194 = Incomplete Adjacency
   11 000011b = 195 = For us

   Examples:

     value : 0x40 = 64
     binary: 01000000
     decode: 01        -> Forward
               000000  -> No further information

     value : 0x89 = 137
     binary: 10001001
     decode: 10        -> Drop
               001001  -> Fragmentation and DF set
```

Abstract Data Type: unsigned32
ElementId: 89
Semantics: identifier
Status: current
Reference:
See [CCO-NF9FMT] - NetFlow Version 9 Record Format.

4.14. srcTrafficIndex

Description:
BGP Policy Accounting Source Traffic Index
Abstract Data Type: unsigned32

ElementId: 92
Semantics: identifier
Status: current
Reference:
BGP policy accounting as described in [CCO-BGPPOL]

4.15. dstTrafficIndex

Description:
BGP Policy Accounting Destination Traffic Index
Abstract Data Type: unsigned32
ElementId: 93
Semantics: identifier
Status: current
Reference:
BGP policy accounting as described in [CCO-BGPPOL]

4.16. className

Description:
Deprecated in favor of 335 selectorName. Traffic Class Name, associated with the classId Information Element.
Abstract Data Type: string
ElementId: 100
Status: deprecated

4.17. layer2packetSectionOffset

Description:
Layer 2 packet section offset. Potentially a generic packet section offset.
Abstract Data Type: unsigned16
ElementId: 102
Semantics: quantity
Status: current
EDITOR’S NOTE: [I-D.kashima-ipfix-data-link-layer-monitoring] contains a corresponding field ’sectionOffset’ with a better description. One solution is to assign the value 102 for the ’sectionOffset’ in [I-D.kashima-ipfix-data-link-layer-monitoring].

4.18. layer2packetSectionSize

Description:
Layer 2 packet section size. Potentially a generic packet section size.

Abstract Data Type: unsigned16
ElementId: 103
Semantics: quantity
Status: current
EDITOR'S NOTE: [I-D.kashima-ipfix-data-link-layer-monitoring] contains a corresponding field 'sectionObservedOctets' with a better description. One solution is to assign the value 103 to 'sectionObservedOctets' in [I-D.kashima-ipfix-data-link-layer-monitoring].

4.19. layer2packetSectionData

Description:
Layer 2 packet section data.
Abstract Data Type: octetArray
ElementId: 104
Status: current
EDITOR's NOTE: [I-D.kashima-ipfix-data-link-layer-monitoring] contains a corresponding field 'dataLinkFrameSection' with a better description. One solution is to assign the value 104 to 'dataLinkFrameSection' in [I-D.kashima-ipfix-data-link-layer-monitoring].

5. Other Information Elements

5.1. Performance Metrics IEs

ElementId: 65 .. 69

Performance metrics will need a consolidation in the industry, based on RFC6390. Once this consolidation happens, via a separate document the IEs 65-69 will either be assigned in the IANA registry or their status will be deprecated.

5.2. Application Information IEs

ElementId: 101

ElementId: 94 .. 97

Please refer to the Export of Application Information in IPFIX [I-D.claise-export-application-info-in-ipfix]

6. IANA Considerations

This document specifies several new IPFIX Information Elements in the

IPFIX Information Element registry as defined in Section 3 above. The following Information Elements must be assigned:

- o IE Number 3 for the deltaFlowCount IE
- o IE Number 34 for the samplingInterval IE
- o IE Number 35 for the samplingAlgorithm IE
- o IE Number 38 for the engineType IE
- o IE Number 39 for the engineId IE
- o IE Number 43 for the ipv4RouterSc IE
- o IE Number 48 for the samplerId IE
- o IE Number 49 for the samplerMode IE
- o IE Number 50 for the samplerRandomInterval IE
- o IE Number 51 for the classId IE
- o IE Number 84 for the samplerName IE
- o IE Number 87 for the flagsAndSamplerId IE
- o IE Number 89 for the forwardingStatus IE
- o IE Number 92 for the srcTrafficIndex IE
- o IE Number 93 for the dstTrafficIndex IE
- o IE Number 100 for the className IE
- o IE Number 102 for the layer2packetSectionOffset IE
- o IE Number 103 for the layer2packetSectionSize IE
- o IE Number 104 for the layer2packetSectionData IE

7. Security Considerations

This document specifies the definitions of several Information Elements and does not alter the security considerations of the base protocol. Please refer to the security considerations sections of RFC 3954 [RFC3954] and RFC 5102 [RFC5102].

However, the export of the sections of the packet payload may unintentionally change the security assumptions of other protocols.

8. References

8.1. Normative References

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

8.2. Informative References

[CCO-BGPPOL]
Cisco, "BGP Policy Accounting and BGP Policy Accounting Output Interface Accounting Features", <http://

www.cisco.com/en/US/tech/tk365/
technologies_tech_note09186a0080094e88.shtml>.

[CCO-MLS] Cisco, "IP MultiLayer Switching Sample Configuration", <http://www.cisco.com/en/US/products/hw/switches/ps700/products_configuration_example09186a00800ab513.shtml>.

[CCO-NF9FMT] Cisco, "NetFlow version 9 Flow-Record format", <http://www.cisco.com/en/US/technologies/tk648/tk362/technologies_white_paper09186a00800a3db9.html>.

[I-D.claise-export-application-info-in-ipfix] Claise, B., Aitken, P., and N. Ben-Dvora, "Export of Application Information in IPFIX", draft-claise-export-application-info-in-ipfix-02 (work in progress), September 2011.

[I-D.ietf-ipfix-flow-selection-tech] D'Antonio, S., Zseby, T., Henke, C., and L. Peluso, "Flow Selection Techniques", draft-ietf-ipfix-flow-selection-tech-06 (work in progress), May 2011.

[I-D.kashima-ipfix-data-link-layer-monitoring] Kashima, S., Nakata, K., and A. Kobayashi, "Information Elements for Data Link Layer Traffic Measurement", draft-kashima-ipfix-data-link-layer-monitoring-06 (work in progress), September 2011.

[I-D.trammell-ipfix-a9n] Trammell, B., Wagner, A., and B. Claise, "Flow Aggregation for the IP Flow Information Export (IPFIX) Protocol", draft-trammell-ipfix-a9n-04 (work in progress), September 2011.

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

[PSAMP-IANA] IANA, "Packet Sampling (PSAMP) Parameters", <http://www.iana.org/assignments/psamp-parameters/psamp-parameters.xml>.

[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.

Appendix A. XML Specification of IPFIX Information Elements


```
<?xml version="1.0" encoding="UTF-8"?>

<fieldDefinitions xmlns="urn:ietf:params:xml:ns:ipfix-info"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:schemaLocation="urn:ietf:params:xml:ns:ipfix-info
                ipfix-info.xsd">

  <field name="deltaFlowCount" dataType="unsigned64"
             group=""
             dataTypeSemantics="quantity"
             elementId="3" applicability="flow" status="current">
    <description>
        <paragraph>
        This Information Element specifies the current number of all
        Flow Records that form the parent population as input to the
        Flow Selection Process.
        </paragraph>
    </description>
  </field>
  <field name="samplingInterval" dataType="unsigned32"
             group=""
             dataTypeSemantics="quantity"
             elementId="34" applicability="flow" status="deprecated">
    <description>
        <paragraph>
        Deprecated in favor of 305 samplingPacketInterval. When using
        sampled NetFlow, the rate at which packets are sampled - e.g. a
        value of 100 indicates that one of every 100 packets is sampled.
        </paragraph>
    </description>
  </field>
  <field name="samplingAlgorithm" dataType="unsigned8"
             group=""
             dataTypeSemantics="identifier"
             elementId="35" applicability="flow" status="deprecated">
    <description>
        <paragraph>
        Deprecated in favor of 304 selectorAlgorithm. The type of
        algorithm used for sampled NetFlow: 1 - Deterministic Sampling;
        2 - Random Sampling. The values are not compatible with the
        selectorAlgorithm IE, where "Deterministic" has been replaced by
```

```
        "Systematic count-based" (1) or "Systematic time-based" (2), and
        "Random" is (3). Conversion is required, see
        <REF:PSAMP-IANA>PSAMP parameters.
        </paragraph>
    </description>
  </field>
  <field name="engineType" dataType="unsigned8"
             group=""
             dataTypeSemantics="identifier"
             elementId="38" applicability="flow" status="deprecated">
    <description>
        <paragraph>
        Type of flow switching engine in a router/switch: RP = 0,
        VIP/Line card = 1, PFC/DFC = 2. Reserved for internal use on the
        collector.
        </paragraph>
    </description>
  </field>
  <field name="engineId" dataType="unsigned8"
             group=""
             dataTypeSemantics="identifier"
             elementId="39" applicability="flow" status="deprecated">
    <description>
        <paragraph>
        VIP or line card slot number of the flow switching engine in a
        router/switch. Reserved for internal use on the collector.
        </paragraph>
    </description>
  </field>
  <field name="ipv4RouterSc" dataType="ipv4Address"
             group=""
             dataTypeSemantics="ipv4Address"
             elementId="43" applicability="flow" status="deprecated">
    <description>
        <paragraph>
        This is a platform-specific field for Catalyst 5000/Catalyst
        6000 family. It is used to store the address of a router that is
        being shortcut when performing MultiLayer Switching.
        </paragraph>
    </description>
    <reference>
        http://www
        .cisco.com
        /en/US/pro
        ducts/hw/s
        witches/ps
        700/products_configuration_example09186a00800ab513.shtml
        describes the MultiLayer Switching.
```

```
    </reference>
  </field>
  <field name="samplerId" dataType="unsigned8"
             group=""
             dataTypeSemantics="identifier"
             elementId="48" applicability="flow" status="deprecated">
    <description>
        <paragraph>
        Deprecated in favor of 302 selectorId. The unique identifier
        associated with samplerName.
        </paragraph>
    </description>
  </field>
  <field name="samplerMode" dataType="unsigned8"
             group=""
             dataTypeSemantics="identifier"
             elementId="49" applicability="flow" status="deprecated">
    <description>
        <paragraph>
        Deprecated in favor of 304 selectorAlgorithm. The values are not
        compatible: selectorAlgorithm=3 is random sampling. The type of
        algorithm used for sampling data: 1 - deterministic, 2 - random
        sampling. Use with samplerRandomInterval.
        </paragraph>
    </description>
  </field>
  <field name="samplerRandomInterval" dataType="unsigned32"
             group=""
             dataTypeSemantics="quantity"
             elementId="50" applicability="flow" status="deprecated">
    <description>
        <paragraph>
        Deprecated in favour of 305 samplingPacketInterval. Packet
        interval at which to sample - in case of random sampling. Used
        in connection with samplerMode 0x02 (random sampling) value.
        </paragraph>
    </description>
  </field>
  <field name="classId" dataType="unsigned8"
             group=""
             dataTypeSemantics="identifier"
             elementId="51" applicability="flow" status="deprecated">
    <description>
        <paragraph>
        Deprecated in favour of 302 selectorId. Characterizes the
        traffic class, i.e. QoS treatment.
        </paragraph>
    </description>
```

```
  </field>
  <field name="samplerName" dataType="string"
             group=""
             dataTypeSemantics=""
             elementId="84" applicability="flow" status="deprecated">
    <description>
        <paragraph>
        Deprecated in favor of 335 selectorName. Name of the flow
        sampler.
        </paragraph>
    </description>
  </field>
  <field name="flagsAndSamplerId" dataType="unsigned32"
             group=""
             dataTypeSemantics="identifier"
             elementId="87" applicability="flow" status="deprecated">
    <description>
        <paragraph>
        Flow flags and the value of the sampler ID (samplerId) combined
        in one bitmapped field. Reserved for internal use on the
        collector.
        </paragraph>
    </description>
  </field>
  <field name="forwardingStatus" dataType="unsigned32"
             group=""
             dataTypeSemantics="identifier"
             elementId="89" applicability="flow" status="current">
    <description>
        <paragraph>
        This Information Element describes the forwarding status of the
        flow and any attached reasons. The Reduced Size Encoding rules
        as per <REF:RFC5101> apply.
        </paragraph>
        <artwork>
           The basic encoding is 8 bits. The future extensions
           could add one or three bytes. The layout of the basic
           encoding is as follows:

              MSB -   0   1   2   3   4   5   6   7   - LSB
                    +---+---+---+---+---+---+---+---+
                    | Status|  Reason code or flags |
                    +---+---+---+---+---+---+---+---+
           Status:

           00b = Unknown
           01b = Forwarded
           10b = Dropped
```

```
          11b = Consumed


          Reason Code (status = 01b, Forwarded)

          01 000000b = 64 = Unknown
          01 000001b = 65 = Fragmented
          01 000010b = 66 = Not Fragmented

          Reason Code (status = 10b, Dropped)

          10 000000b = 128 = Unknown
          10 000001b = 129 = ACL deny
          10 000010b = 130 = ACL drop
          10 000011b = 131 = Unroutable
          10 000100b = 132 = Adjacency
          10 000101b = 133 = Fragmentation and DF set
          10 000110b = 134 = Bad header checksum
          10 000111b = 135 = Bad total Length
          10 001000b = 136 = Bad header length
          10 001001b = 137 = bad TTL
          10 001010b = 138 = Policer
          10 001011b = 139 = WRED
          10 001100b = 140 = RPF
          10 001101b = 141 = For us
          10 001110b = 142 = Bad output interface
          10 001111b = 143 = Hardware

          Reason Code (status = 11b, Consumed)

          11 000000b = 192 = Unknown
          11 000001b = 193 = Punt Adjacency
          11 000010b = 194 = Incomplete Adjacency
          11 000011b = 195 = For us

          Examples:

            value : 0x40 = 64
            binary: 01000000
            decode: 01        -> Forward
                      000000  -> No further information

            value : 0x89 = 137
            binary: 10001001
            decode: 10        -> Drop
                      001001  -> Fragmentation and DF set
        </artwork>
    </description>
```

```
    <reference>
        See http:/
        /www.cisco
        .com/en/US
        /technolog
        ies/tk648/tk362/technologies_white_paper09186a00800a3db9.html -
        NetFlow Version 9 Record Format.
    </reference>
  </field>
  <field name="srcTrafficIndex" dataType="unsigned32"
              group=""
              dataTypeSemantics="identifier"
              elementId="92" applicability="flow" status="current">
    <description>
        <paragraph>
        BGP Policy Accounting Source Traffic Index
        </paragraph>
    </description>
    <reference>
        BGP policy accounting as described in
        http://www
        .cisco.com
        /en/US/tech/tk365/technologies_tech_note09186a0080094e88.shtml
    </reference>
  </field>
  <field name="dstTrafficIndex" dataType="unsigned32"
              group=""
              dataTypeSemantics="identifier"
              elementId="93" applicability="flow" status="current">
    <description>
        <paragraph>
        BGP Policy Accounting Destination Traffic Index
        </paragraph>
    </description>
    <reference>
        BGP policy accounting as described in
        http://www
        .cisco.com
        /en/US/tech/tk365/technologies_tech_note09186a0080094e88.shtml
    </reference>
  </field>
  <field name="className" dataType="string"
              group=""
              dataTypeSemantics=""
              elementId="100" applicability="flow" status="deprecated">
    <description>
        <paragraph>
        Deprecated in favor of 335 selectorName. Traffic Class Name,
```

```
        associated with the classId Information Element.
        </paragraph>
    </description>
  </field>
  <field name="layer2packetSectionOffset" dataType="unsigned16"
             group=""
             dataTypeSemantics="quantity"
             elementId="102" applicability="flow" status="current">
    <description>
        <paragraph>
        Layer 2 packet section offset. Potentially a generic packet
        section offset.
        </paragraph>
    </description>
  </field>
  <field name="layer2packetSectionSize" dataType="unsigned16"
             group=""
             dataTypeSemantics="quantity"
             elementId="103" applicability="flow" status="current">
    <description>
        <paragraph>
        Layer 2 packet section size. Potentially a generic packet
        section size.
        </paragraph>
    </description>
  </field>
  <field name="layer2packetSectionData" dataType="octetArray"
             group=""
             dataTypeSemantics=""
             elementId="104" applicability="flow" status="current">
    <description>
        <paragraph>
        Layer 2 packet section data.
        </paragraph>
    </description>
  </field>
</fieldDefinitions>
```


Appendix B. Changes

To be removed by RFC Editor before publication

01: initial revision presented at the IETF meeting.

02: removed "flow" from flowSamplerId, flowSamplerMode, and flowSamplerRandomInterval; updated the related drafts in references;

added the "reference" column to the XML definitions; renamed fsFlowEntryTotalCount into deltaFlowCount to keep the naming in sync with [I-D.trammell-ipfix-a9n]. Also minor changes to formatting and added the IE overview table.

Authors' Addresses

Andrew Yourtchenko
Cisco Systems, Inc.
De Kleetlaan, 7
Brussels, Diegem B-1831
Belgium

Phone: +32 2 704 5494
Email: ayourtch@cisco.com

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

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

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

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