

Network Working Group  
Request for Comments: 5102  
Category: Standards Track

J. Quittek  
NEC  
S. Bryant  
B. Claise  
P. Aitken  
Cisco Systems, Inc.  
J. Meyer  
PayPal  
January 2008

## Information Model for IP Flow Information Export

### Status of This Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

### 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.

### Table of Contents

1. Introduction .....	6
2. Properties of IPFIX Protocol Information Elements .....	7
2.1. Information Elements Specification Template .....	7
2.2. Scope of Information Elements .....	9
2.3. Naming Conventions for Information Elements .....	9
3. Type Space .....	10
3.1. Abstract Data Types .....	10
3.1.1. unsigned8 .....	10
3.1.2. unsigned16 .....	11
3.1.3. unsigned32 .....	11
3.1.4. unsigned64 .....	11
3.1.5. signed8 .....	11
3.1.6. signed16 .....	11
3.1.7. signed32 .....	11
3.1.8. signed64 .....	11

3.1.9.	float32	11
3.1.10.	float64	11
3.1.11.	boolean	12
3.1.12.	macAddress	12
3.1.13.	octetArray	12
3.1.14.	string	12
3.1.15.	dateTimeSeconds	12
3.1.16.	dateTimeMilliseconds	12
3.1.17.	dateTimeMicroseconds	12
3.1.18.	dateTimeNanoseconds	13
3.1.19.	ipv4Address	13
3.1.20.	ipv6Address	13
3.2.	Data Type Semantics	13
3.2.1.	quantity	13
3.2.2.	totalCounter	13
3.2.3.	deltaCounter	14
3.2.4.	identifier	14
3.2.5.	flags	14
4.	Information Element Identifiers	14
5.	Information Elements	18
5.1.	Identifiers	19
5.1.1.	lineCardId	20
5.1.2.	portId	20
5.1.3.	ingressInterface	20
5.1.4.	egressInterface	21
5.1.5.	meteringProcessId	21
5.1.6.	exportingProcessId	21
5.1.7.	flowId	22
5.1.8.	templateId	22
5.1.9.	observationDomainId	22
5.1.10.	observationPointId	23
5.1.11.	commonPropertiesId	23
5.2.	Metering and Exporting Process Configuration	23
5.2.1.	exporterIPv4Address	24
5.2.2.	exporterIPv6Address	24
5.2.3.	exporterTransportPort	24
5.2.4.	collectorIPv4Address	25
5.2.5.	collectorIPv6Address	25
5.2.6.	exportInterface	25
5.2.7.	exportProtocolVersion	26
5.2.8.	exportTransportProtocol	26
5.2.9.	collectorTransportPort	27
5.2.10.	flowKeyIndicator	27
5.3.	Metering and Exporting Process Statistics	28
5.3.1.	exportedMessageTotalCount	28
5.3.2.	exportedOctetTotalCount	28
5.3.3.	exportedFlowRecordTotalCount	29
5.3.4.	observedFlowTotalCount	29

5.3.5.	ignoredPacketTotalCount	29
5.3.6.	ignoredOctetTotalCount	30
5.3.7.	notSentFlowTotalCount	30
5.3.8.	notSentPacketTotalCount	30
5.3.9.	notSentOctetTotalCount	31
5.4.	IP Header Fields	31
5.4.1.	ipVersion	31
5.4.2.	sourceIPv4Address	32
5.4.3.	sourceIPv6Address	32
5.4.4.	sourceIPv4PrefixLength	32
5.4.5.	sourceIPv6PrefixLength	33
5.4.6.	sourceIPv4Prefix	33
5.4.7.	sourceIPv6Prefix	33
5.4.8.	destinationIPv4Address	33
5.4.9.	destinationIPv6Address	34
5.4.10.	destinationIPv4PrefixLength	34
5.4.11.	destinationIPv6PrefixLength	34
5.4.12.	destinationIPv4Prefix	34
5.4.13.	destinationIPv6Prefix	35
5.4.14.	ipTTL	35
5.4.15.	protocolIdentifier	35
5.4.16.	nextHeaderIPv6	36
5.4.17.	ipDiffServCodePoint	36
5.4.18.	ipPrecedence	36
5.4.19.	ipClassOfService	37
5.4.20.	postIpClassOfService	37
5.4.21.	flowLabelIPv6	38
5.4.22.	isMulticast	38
5.4.23.	fragmentIdentification	39
5.4.24.	fragmentOffset	39
5.4.25.	fragmentFlags	39
5.4.26.	ipHeaderLength	40
5.4.27.	ipv4IHL	40
5.4.28.	totalLengthIPv4	41
5.4.29.	ipTotalLength	41
5.4.30.	payloadLengthIPv6	41
5.5.	Transport Header Fields	42
5.5.1.	sourceTransportPort	42
5.5.2.	destinationTransportPort	42
5.5.3.	udpSourcePort	43
5.5.4.	udpDestinationPort	43
5.5.5.	udpMessageLength	43
5.5.6.	tcpSourcePort	44
5.5.7.	tcpDestinationPort	44
5.5.8.	tcpSequenceNumber	44
5.5.9.	tcpAcknowledgementNumber	44
5.5.10.	tcpWindowSize	45
5.5.11.	tcpWindowScale	45

5.5.12.	tcpUrgentPointer	45
5.5.13.	tcpHeaderLength	45
5.5.14.	icmpTypeCodeIPv4	46
5.5.15.	icmpTypeIPv4	46
5.5.16.	icmpCodeIPv4	46
5.5.17.	icmpTypeCodeIPv6	46
5.5.18.	icmpTypeIPv6	47
5.5.19.	icmpCodeIPv6	47
5.5.20.	igmpType	47
5.6.	Sub-IP Header Fields	48
5.6.1.	sourceMacAddress	48
5.6.2.	postSourceMacAddress	48
5.6.3.	vlanId	49
5.6.4.	postVlanId	49
5.6.5.	destinationMacAddress	49
5.6.6.	postDestinationMacAddress	49
5.6.7.	wlanChannelId	50
5.6.8.	wlanSSID	50
5.6.9.	mplsTopLabelTTL	50
5.6.10.	mplsTopLabelExp	51
5.6.11.	postMplsTopLabelExp	51
5.6.12.	mplsLabelStackDepth	51
5.6.13.	mplsLabelStackLength	52
5.6.14.	mplsPayloadLength	52
5.6.15.	mplsTopLabelStackSection	52
5.6.16.	mplsLabelStackSection2	53
5.6.17.	mplsLabelStackSection3	53
5.6.18.	mplsLabelStackSection4	53
5.6.19.	mplsLabelStackSection5	54
5.6.20.	mplsLabelStackSection6	54
5.6.21.	mplsLabelStackSection7	54
5.6.22.	mplsLabelStackSection8	55
5.6.23.	mplsLabelStackSection9	55
5.6.24.	mplsLabelStackSection10	55
5.7.	Derived Packet Properties	56
5.7.1.	ipPayloadLength	56
5.7.2.	ipNextHopIPv4Address	56
5.7.3.	ipNextHopIPv6Address	57
5.7.4.	bgpSourceAsNumber	57
5.7.5.	bgpDestinationAsNumber	57
5.7.6.	bgpNextAdjacentAsNumber	57
5.7.7.	bgpPrevAdjacentAsNumber	58
5.7.8.	bgpNextHopIPv4Address	58
5.7.9.	bgpNextHopIPv6Address	58
5.7.10.	mplsTopLabelType	59
5.7.11.	mplsTopLabelIPv4Address	59
5.7.12.	mplsTopLabelIPv6Address	60
5.7.13.	mplsVpnRouteDistinguisher	60

5.8. Min/Max Flow Properties .....	61
5.8.1. minimumIpTotalLength .....	61
5.8.2. maximumIpTotalLength .....	61
5.8.3. minimumTTL .....	61
5.8.4. maximumTTL .....	62
5.8.5. ipv4Options .....	62
5.8.6. ipv6ExtensionHeaders .....	64
5.8.7. tcpControlBits .....	65
5.8.8. tcpOptions .....	66
5.9. Flow Timestamps .....	67
5.9.1. flowStartSeconds .....	67
5.9.2. flowEndSeconds .....	68
5.9.3. flowStartMilliseconds .....	68
5.9.4. flowEndMilliseconds .....	68
5.9.5. flowStartMicroseconds .....	68
5.9.6. flowEndMicroseconds .....	68
5.9.7. flowStartNanoseconds .....	69
5.9.8. flowEndNanoseconds .....	69
5.9.9. flowStartDeltaMicroseconds .....	69
5.9.10. flowEndDeltaMicroseconds .....	69
5.9.11. systemInitTimeMilliseconds .....	70
5.9.12. flowStartSysUpTime .....	70
5.9.13. flowEndSysUpTime .....	70
5.10. Per-Flow Counters .....	70
5.10.1. octetDeltaCount .....	71
5.10.2. postOctetDeltaCount .....	71
5.10.3. octetDeltaSumOfSquares .....	72
5.10.4. octetTotalCount .....	72
5.10.5. postOctetTotalCount .....	72
5.10.6. octetTotalSumOfSquares .....	72
5.10.7. packetDeltaCount .....	73
5.10.8. postPacketDeltaCount .....	73
5.10.9. packetTotalCount .....	73
5.10.10. postPacketTotalCount .....	74
5.10.11. droppedOctetDeltaCount .....	74
5.10.12. droppedPacketDeltaCount .....	74
5.10.13. droppedOctetTotalCount .....	74
5.10.14. droppedPacketTotalCount .....	75
5.10.15. postMCastPacketDeltaCount .....	75
5.10.16. postMCastOctetDeltaCount .....	75
5.10.17. postMCastPacketTotalCount .....	76
5.10.18. postMCastOctetTotalCount .....	76
5.10.19. tcpSynTotalCount .....	76
5.10.20. tcpFinTotalCount .....	77
5.10.21. tcpRstTotalCount .....	77
5.10.22. tcpPshTotalCount .....	77
5.10.23. tcpAckTotalCount .....	78
5.10.24. tcpUrgTotalCount .....	78

5.11. Miscellaneous Flow Properties .....	78
5.11.1. flowActiveTimeout .....	79
5.11.2. flowIdleTimeout .....	79
5.11.3. flowEndReason .....	79
5.11.4. flowDurationMilliseconds .....	80
5.11.5. flowDurationMicroseconds .....	80
5.11.6. flowDirection .....	80
5.12. Padding .....	80
5.12.1. paddingOctets .....	81
6. Extending the Information Model .....	81
7. IANA Considerations .....	82
7.1. IPFIX Information Elements .....	82
7.2. MPLS Label Type Identifier .....	82
7.3. XML Namespace and Schema .....	83
8. Security Considerations .....	83
9. Acknowledgements .....	84
10. References .....	84
10.1. Normative References .....	84
10.2. Informative References .....	84
Appendix A. XML Specification of IPFIX Information Elements .....	88
Appendix B. XML Specification of Abstract Data Types .....	157

## 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 [RFC5101] 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 [RFC5101].

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 [RFC5101]. As in [RFC5101], 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].

## 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 [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.

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 [RFC5101].

## 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 classOfServiceIPv4. Then the value observed and reported by Information Element classOfServiceIPv4 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, "postClassOfServiceIPv4". 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 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.

### 3.1.16. dateTimeMilliseconds

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.

### 3.1.17. dateTimeMicroseconds

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.

#### 3.1.18. dateTimeNanoseconds

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.

#### 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, "postClassOfServiceIPv4", 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.

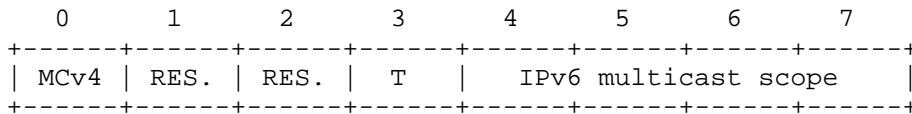
## 5.4.22. isMulticast

## 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.



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 source 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.

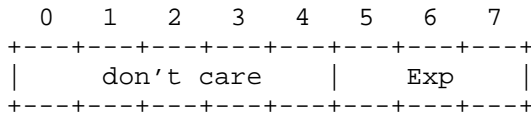
## 5.6.10. mplsTopLabelExp

## 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.



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

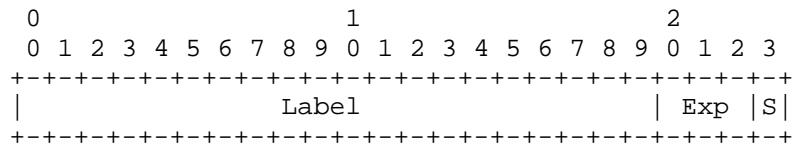
Reference:

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.



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:

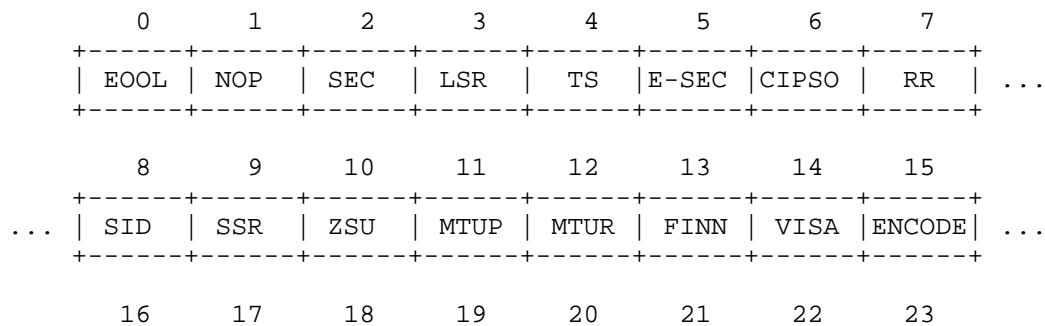
See RFC 791 for the definition of the IPv4 Time to Live field.

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.



```

+-----+-----+-----+-----+-----+-----+-----+-----+
... | IMITD | EIP  | TR  |ADDEXT|RTRALT| SDB  |NSAPA | DPS  | ...
+-----+-----+-----+-----+-----+-----+-----+-----+
      24      25      26      27      28      29      30      31
+-----+-----+-----+-----+-----+-----+-----+-----+
... | UMP  | QS   |   to be assigned by IANA   | EXP  |   |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Bit	Type Value	Option 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

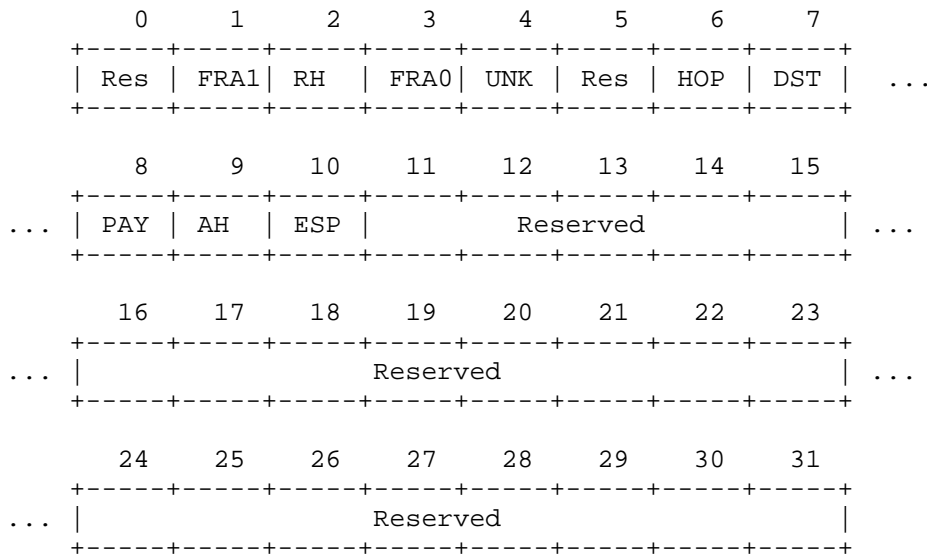
Reference:

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.



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:

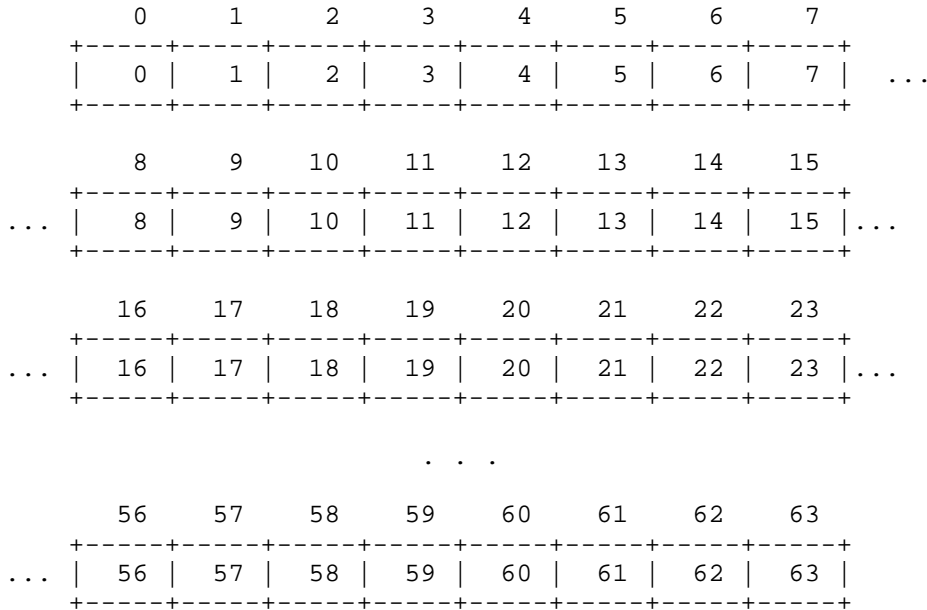
See RFC 793 for the definition of the TCP control bits in the TCP header.

## 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.



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	<code>flowStartSeconds</code>	156	<code>flowStartNanoseconds</code>
151	<code>flowEndSeconds</code>	157	<code>flowEndNanoseconds</code>
152	<code>flowStartMilliseconds</code>	158	<code>flowStartDeltaMicroseconds</code>
153	<code>flowEndMilliseconds</code>	159	<code>flowEndDeltaMicroseconds</code>
154	<code>flowStartMicroseconds</code>	160	<code>systemInitTimeMilliseconds</code>
155	<code>flowEndMicroseconds</code>	22	<code>flowStartSysUpTime</code>
		21	<code>flowEndSysUpTime</code>

### 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 [RFC2434], 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 [RFC2434], 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-15
  - \* 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-15
  - \* 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 [RFC5101].

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.
- [RFC5101] Claise, B., "Specification of the IPFIX Protocol for the Exchange", RFC 5101, January 2008.

### 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.

- [RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.
- [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.
- [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.

## 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 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.
  </paragraph>
</reference>
```

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.
    </paragraph>
  </description>
  <units>packets</units>
</field>
```

```

    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
  +-----+-----+-----+-----+-----+-----+-----+
  | MCv4 | RES. | RES. | T   |   IPv6 multicast scope   |

```

```

+-----+-----+-----+-----+-----+-----+-----+-----+
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>
</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>
```

```
</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
      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="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.
    </paragraph>
  </reference>
</field>
```

```
</paragraph>
</reference>
</field>

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

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

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



```
<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>
```

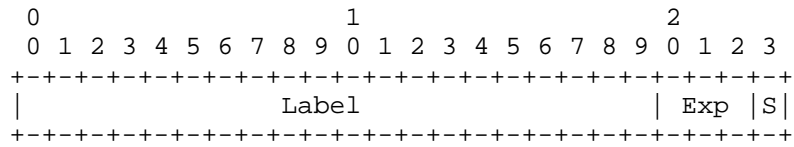
```

<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>

```



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>
```



```
</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.
</reference>
</field>
```

```
</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.
  </paragraph>
</reference>
</field>

<field name="mplsTopLabelIPv6Address" dataType="ipv6Address"
  group="derived"
  dataTypeSemantics="identifier"
  elementId="140" applicability="data" status="current">
  <description>
    <paragraph>
      The IPv6 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.
    </paragraph>
  </reference>
</field>

<field name="mplsVpnRouteDistinguisher" dataType="octetArray"
  group="derived"
  dataTypeSemantics="identifier"
  elementId="90" applicability="all" status="current">
  <description>
    <paragraph>
      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.
    </paragraph>
  </description>
  <reference>
    <paragraph>
      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.
</paragraph>
</reference>
</field>

<field name="minimumIpTotalLength" dataType="unsigned64"
      group="minMax"
      elementId="25" applicability="all" status="current">
  <description>
    <paragraph>
      Length of the smallest packet observed for this Flow.
      The packet length includes the IP header(s) length and
      the IP payload length.
    </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="maximumIpTotalLength" dataType="unsigned64"
      group="minMax"
      elementId="26" applicability="all" status="current">
  <description>
    <paragraph>
      Length of the largest packet observed for this Flow.
      The packet length includes the IP header(s) length and
      the IP payload length.
    </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>
```

```
</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>

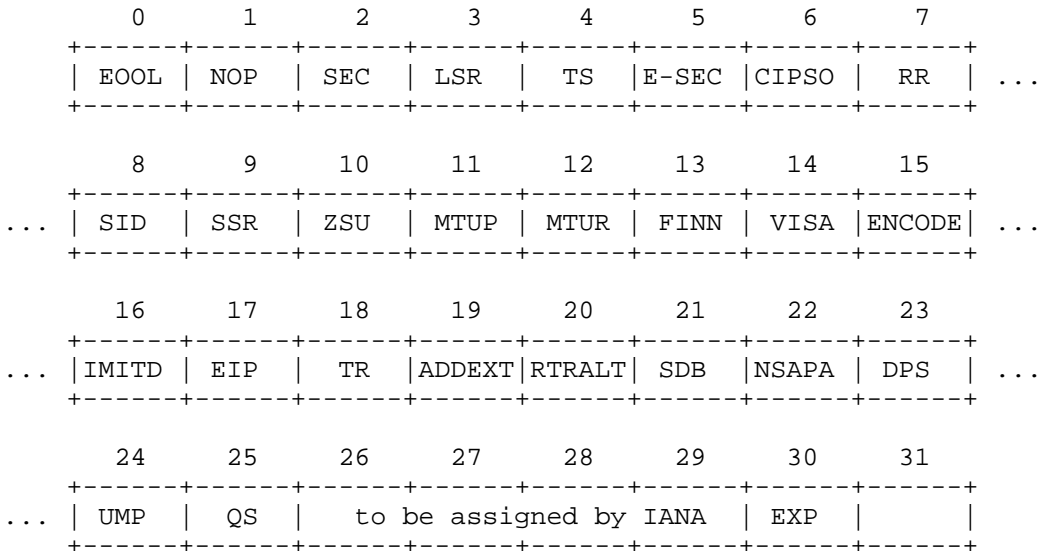
<paragraph>

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>



Bit Value	Type	Option 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>

```

```

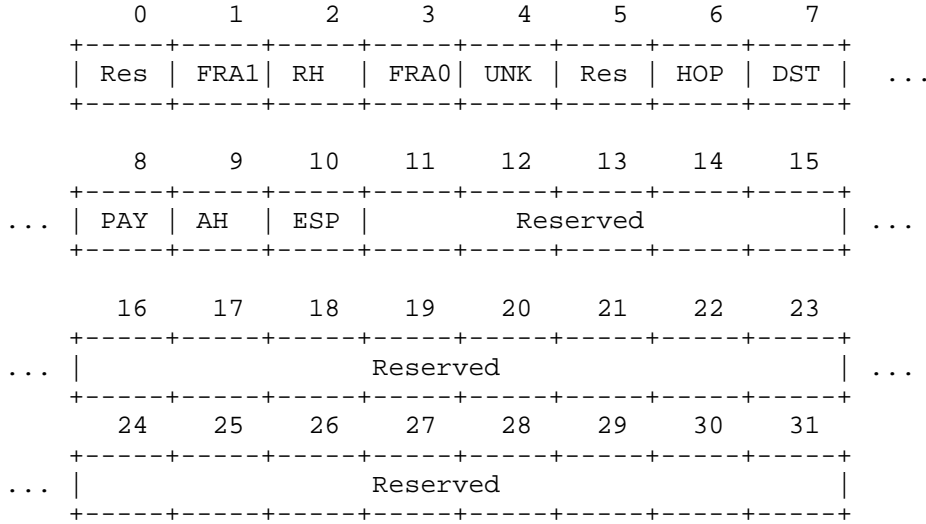
<field name="ipv6ExtensionHeaders" dataType="unsigned32"
  dataTypeSemantics="flags"
  group="minMax"
  elementId="64" applicability="all" status="current">
  <description>
    <paragraph>

```

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>



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
+-----+-----+-----+-----+-----+-----+-----+
|  0  |  1  |  2  |  3  |  4  |  5  |  6  |  7  | ...
+-----+-----+-----+-----+-----+-----+-----+
      8      9     10     11     12     13     14     15
... |  8  |  9  | 10  | 11  | 12  | 13  | 14  | 15 | ...
+-----+-----+-----+-----+-----+-----+-----+
      16     17     18     19     20     21     22     23
... | 16  | 17  | 18  | 19  | 20  | 21  | 22  | 23 | ...
+-----+-----+-----+-----+-----+-----+-----+
      . . .

      56     57     58     59     60     61     62     63
... | 56  | 57  | 58  | 59  | 60  | 61  | 62  | 63 |
+-----+-----+-----+-----+-----+-----+-----+
</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">
<description>
<paragraph>
The absolute timestamp of the last (re-)initialization of the
IPFIX Device.
</paragraph>
</description>
<units>milliseconds</units>
</field>
```

```
<field name="flowStartSysUpTime" dataType="unsigned32"
  group="timestamp"
  elementId="22" applicability="data" status="current">
  <description>
    <paragraph>
      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).
    </paragraph>
  </description>
  <units>milliseconds</units>
</field>

<field name="flowEndSysUpTime" dataType="unsigned32"
  group="timestamp"
  elementId="21" applicability="data" status="current">
  <description>
    <paragraph>
      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).
    </paragraph>
  </description>
  <units>milliseconds</units>
</field>

<field name="octetDeltaCount" dataType="unsigned64"
  dataTypeSemantics="deltaCounter"
  group="flowCounter"
  elementId="1" applicability="data" status="current">
  <description>
    <paragraph>
      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.
    </paragraph>
  </description>
  <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>
```

```
</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>
```

```
</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>
    </restriction>
  </simpleType>

```

```
<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>
```

```
    </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>
  </restriction>
</simpleType>
```

```

    </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.
    </documentation>
  </annotation>
</attribute>
```



```
        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](mailto:stbryant@cisco.com)

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

Phone: +32 2 704 5622  
EMail: [bclaise@cisco.com](mailto:bclaise@cisco.com)

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

Phone: +44 131 561 3616  
EMail: [paitken@cisco.com](mailto: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](mailto:jemeyer@paypal.com)  
URI: <http://www.paypal.com>

## Full Copyright Statement

Copyright (C) The IETF Trust (2008).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

## Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at [ietf-ipr@ietf.org](mailto:ietf-ipr@ietf.org).

