Network Working Group
Internet Draft
Obsoletes: 5101
Category: Standards Track
Expires: December 15, 2013

B. Claise, Ed.
Cisco Systems, Inc.
B. Trammell, Ed.
ETH Zurich

June 13, 2013

Specification of the IP Flow Information eXport (IPFIX) Protocol for the Exchange of Flow Information
draft-ietf-ipfix-protocol-rfc5101bis-08

Abstract

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

Status of This Memo

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

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

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

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

Copyright Notice

Table of Contents

1. Introduction

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

1.1. Changes since RFC 5101

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

- All outstanding technical and editorial errata on [RFC5101] have been addressed.

- The encoding of the dateTimeSeconds, dateTimeMilliseconds, dateTimeMicroseconds, and dateTimeNanoseconds data types, and the related encoding of the IPFIX Message Header Export Time field, have been clarified, especially with respect to the epoch upon which the timestamp data types are based.

- A new Section 5.2 has been added to address wraparound of these timestamp data types after they overflow in 2032 - 2038 CE (common era).

- Clarifications on encoding, especially in Section 6: all IPFIX values are encoded big-endian.

- Template management in section 8 has been simplified and clarified: the specification has been relaxed with respect to [RFC5101], especially concerning potential failures in Template ID reuse. Additional corner cases in template management have been addressed. The new template management language is interoperable with that in [RFC5101] to the extent that the behavior was defined in the prior specification.

- Section 11.3 (Mutual Authentication) has been improved to refer to current practices in TLS mutual authentication; references to Punycode were removed as these are covered in [RFC6125].

- Editorial improvements, including structural changes to sections 8, 9, and 10 to improve readability. Behavior common to all transport protocols has been separated out, with exceptions per transport specifically noted. All template management language (on both Exporting and Collecting Processes) has been unified in section 8.

1.2. IPFIX Documents Overview

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

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

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

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

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

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

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

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

2. Terminology

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

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

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

Observation Point

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

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

Observation Domain

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

Traffic Flow or Flow

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

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

1. one or more packet header fields (e.g., destination IP address), transport header fields (e.g., destination port number), or application header fields (e.g., RTP header fields [RFC3550]).

2. one or more characteristics of the packet itself (e.g., number of MPLS labels, etc...).

3. one or more of fields derived from packet treatment (e.g., next hop IP address, the output interface, etc...).

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

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

Flow Key

Each of the fields that:

1. belong to the packet header (e.g., destination IP address), or

2. are a property of the packet itself (e.g., packet length), or

3. are derived from packet treatment (e.g., Autonomous System (AS) number),

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

Flow Record

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

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

Metering Process

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

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

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

Exporting Process

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

Exporter

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

IPFIX Device

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

Collecting Process

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

Collector

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

Template

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

IPFIX Message

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

Message Header

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

Template Record

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

Data Record

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

Options Template Record

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

Set

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

Template Set

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

Options Template Set

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

Data Set

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

Information Element

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

Transport Session

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

2.1. Terminology Summary Table

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

Figure A: Terminology Summary Table

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

A Template Set contains only Template Record(s).

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

3. IPFIX Message Format

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

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

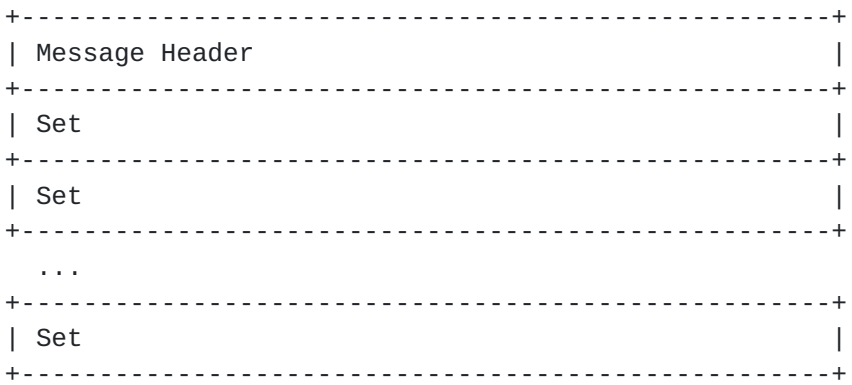


Figure B: IPFIX Message Format

Following are some examples of IPFIX Messages:

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

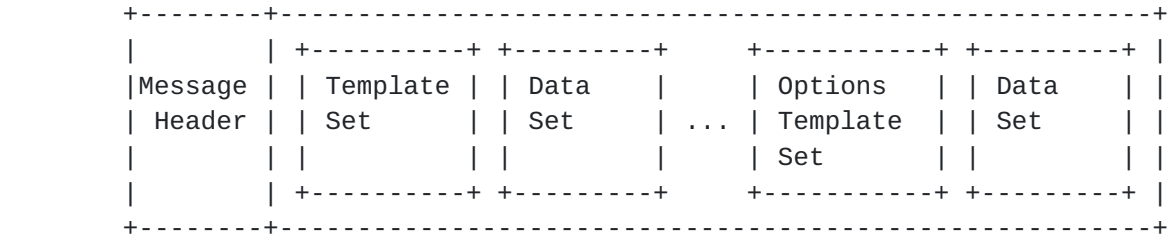


Figure C: IPFIX Message, Example 1

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

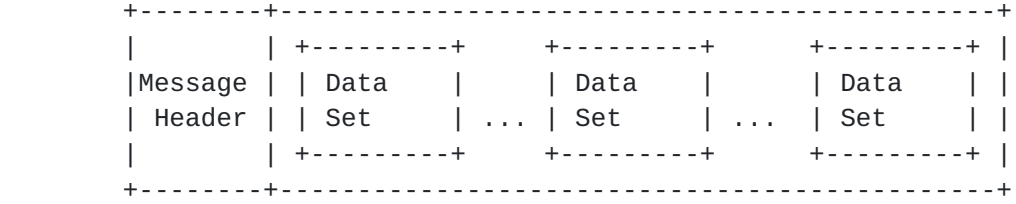


Figure D: IPFIX Message, Example 2

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

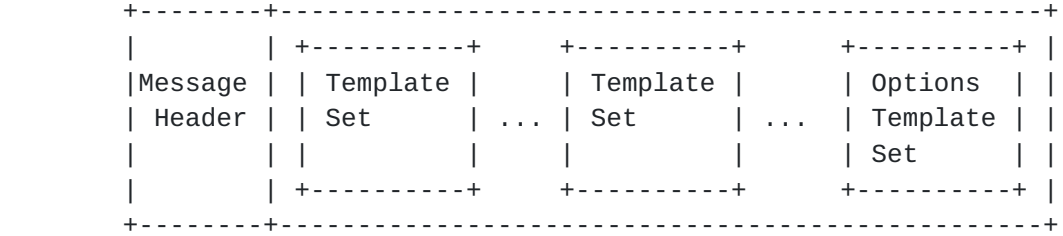


Figure E: IPFIX Message, Example 3

3.1. Message Header Format

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

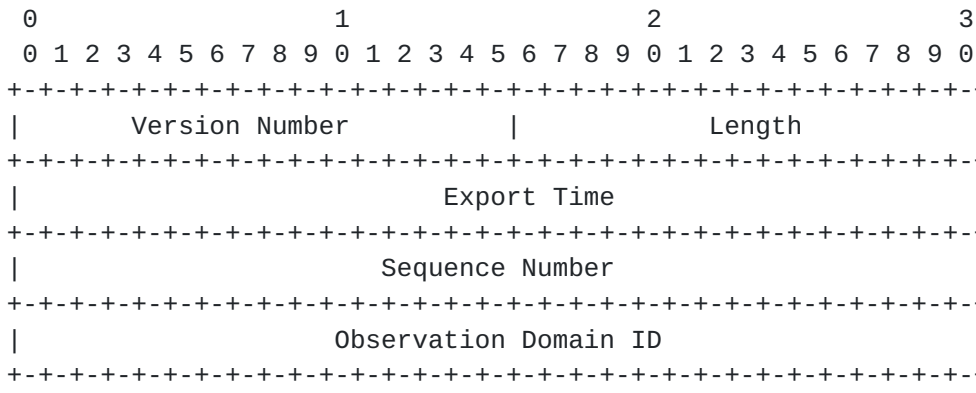


Figure F: IPFIX Message Header Format

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

Version

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

Length

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

Export Time

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

Sequence Number

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

Observation Domain ID

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

3.2. Field Specifier Format

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

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

The Field Specifier format is shown in Figure G.

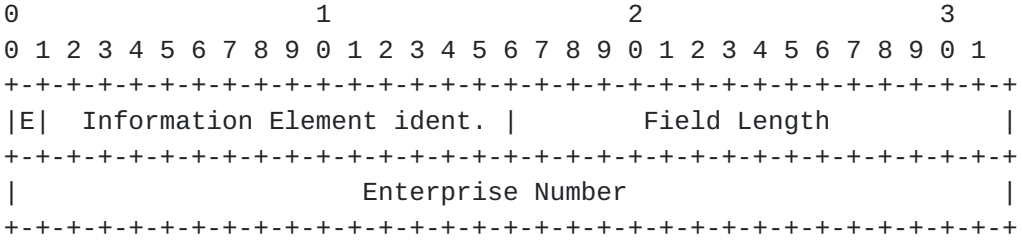


Figure G: Field Specifier Format

Where:

E

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

Information Element identifier

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

Field Length

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

Enterprise Number

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

3.3. Set and Set Header Format

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

3.3.1. Set Format

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

Figure H: Set Format

Set Header

The Set Header Format is defined in Section 3.3.2.

Record

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

Padding

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

3.3.2. Set Header Format

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

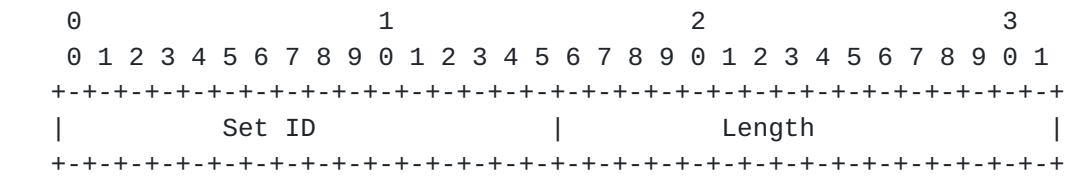


Figure I: Set Header Format

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

Set ID

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

Length

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

3.4. Record Format

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

3.4.1. Template Record Format

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

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

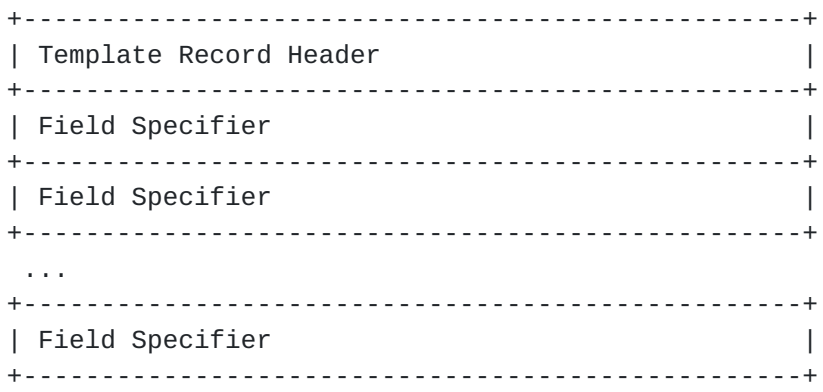


Figure J: Template Record Format

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

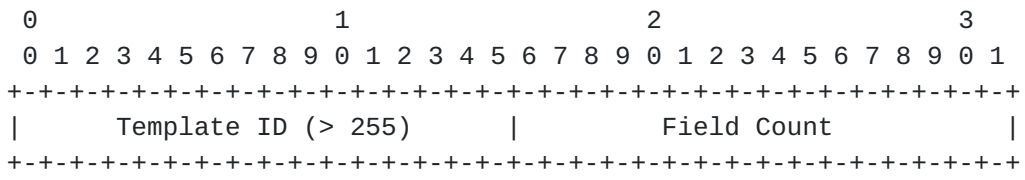


Figure K: Template Record Header Format

The Template Record Header Field Definitions are as follows:

Template ID

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

Field Count

Number of fields in this Template Record.

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

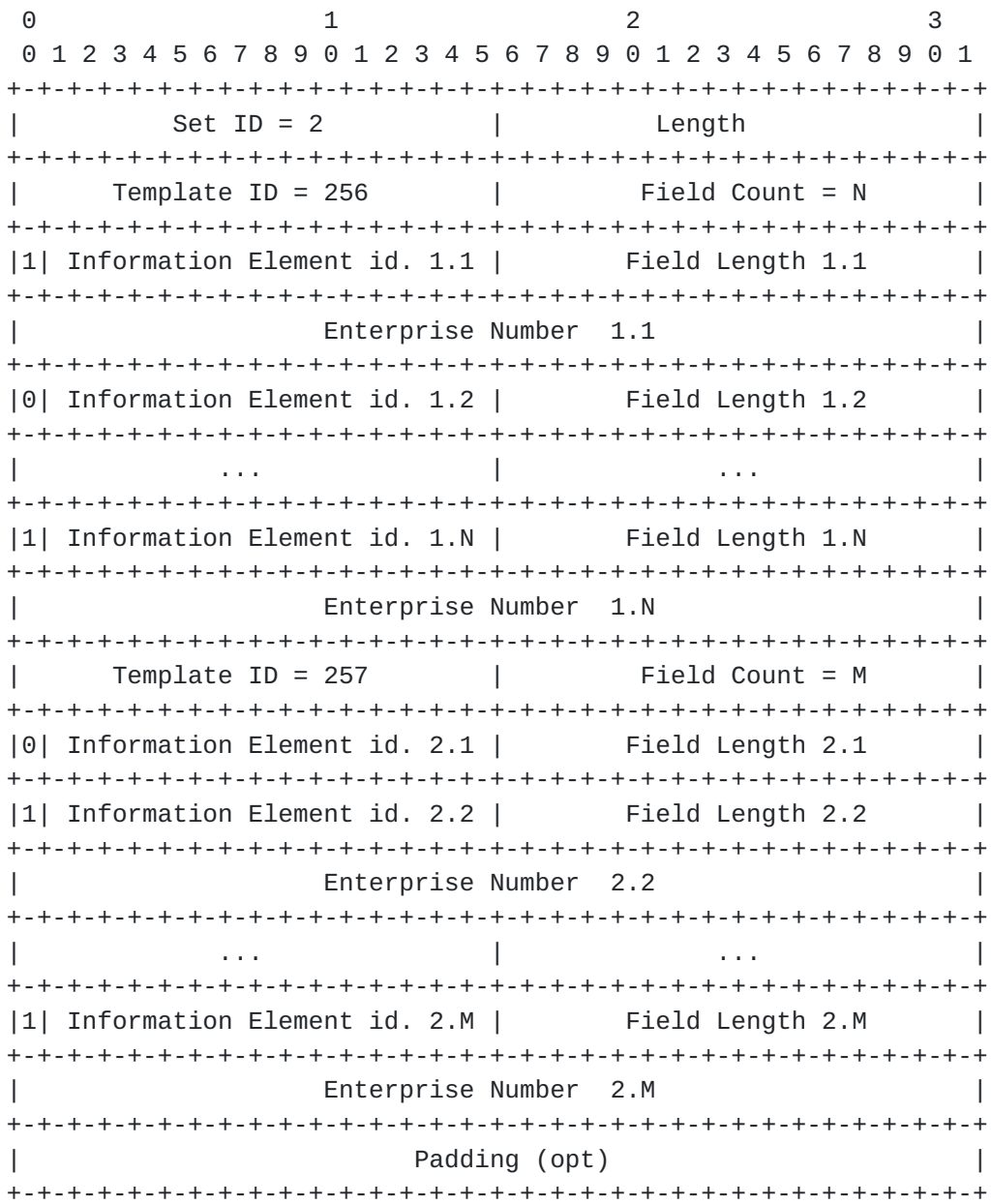


Figure L: Template Set Example

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

3.4.2. Options Template Record Format

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

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

3.4.2.1. Scope

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

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

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

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

3.4.2.2. Options Template Record Format

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

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

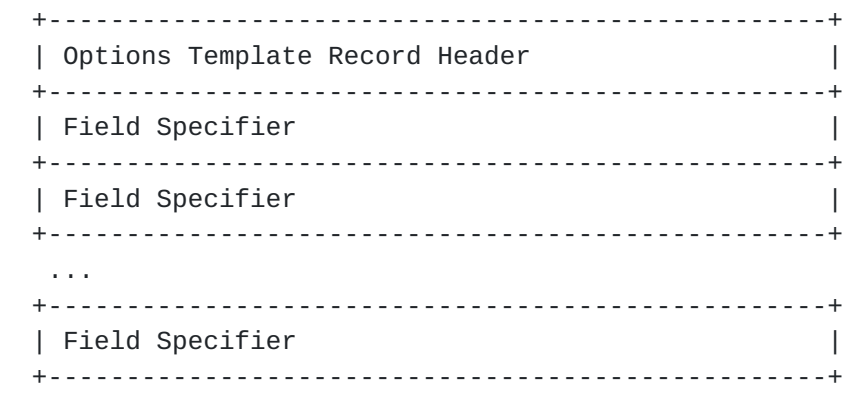


Figure M: Options Template Record Format

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

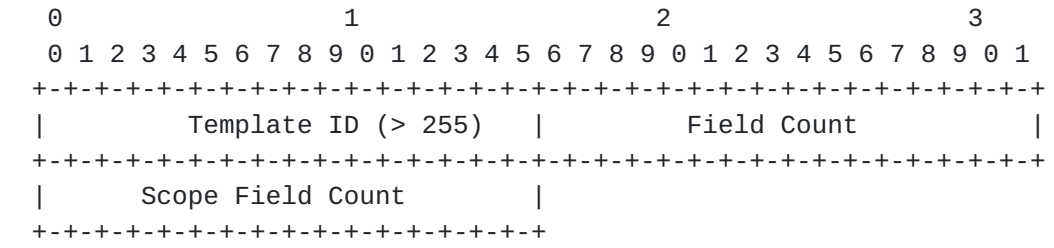


Figure N: Options Template Record Header Format

The Options Template Record Header Field Definitions are as follows:

Template ID

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

Field Count

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

Scope Field Count

Number of scope fields in this Options Template Record. The Scope Fields are normal Fields except that they are interpreted as scope at the Collector. A scope field count of N specifies that the first N Field Specifiers in the Template Record are Scope Fields. The Scope Field Count MUST NOT be zero.

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

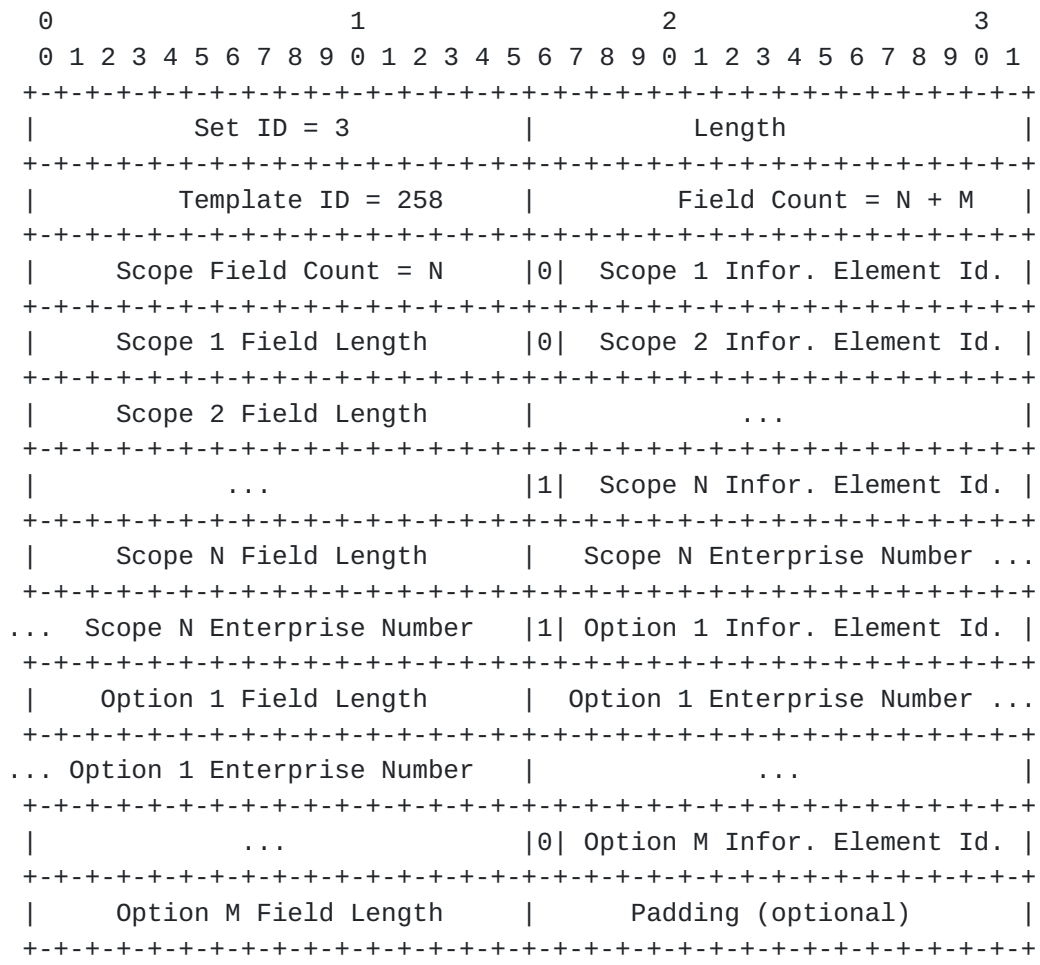


Figure O: Options Template Set Example

3.4.3. Data Record Format

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

Figure P: Data Record Format

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

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

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

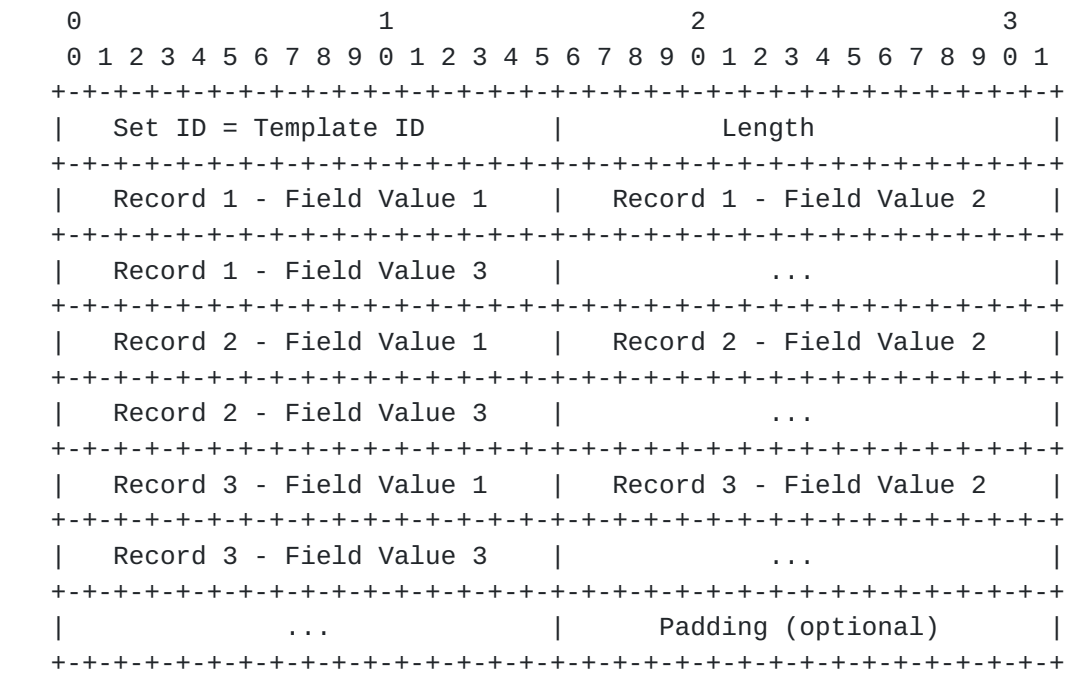


Figure Q: Data Set, containing Data Records

4. Specific Reporting Requirements

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

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

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

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

4.1. The Metering Process Statistics Options Template

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

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

(scope) meteringProcessId
If present, this Information Element MUST be defined as a Scope Field.

exportedMessageTotalCount

exportedFlowRecordTotalCount

exportedOctetTotalCount

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

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

4.2. The Metering Process Reliability Statistics Options Template

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

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

(scope) meteringProcessId
If present, this Information Element MUST be defined as a Scope Field.

ignoredPacketTotalCount

ignoredOctetTotalCount

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

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

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

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

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

4.3. The Exporting Process Reliability Statistics Options Template

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

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

notSentFlowTotalCount

notSentPacketTotalCount

notSentOctetTotalCount

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

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

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

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

4.4. The Flow Keys Options Template

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

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

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

5. Timing Considerations

5.1 IPFIX Message Header Export Time and Flow Record Time

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

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

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

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

5.2 Supporting Timestamp Wraparound

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

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

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

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

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

6. Linkage with the Information Model

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

6.1. Encoding of IPFIX Data Types

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

6.1.1. Integral Data Types

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

6.1.2. Address Types

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

6.1.3. float32

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

6.1.4. float64

The float64 data type MUST be encoded as an IEEE double-precision 64-bit floating point-type, as specified in [IEEE.754.1985], in network byte order.

6.1.5. boolean

The boolean data type is specified according to the TruthValue in [RFC2579]. It is encoded as a single-octet integer, as in Section 6.1.1., with the value 1 for true and a value 2 for false. Every other value is undefined.

6.1.6. string and octetArray

The data type string represents a finite length string of valid characters of the Unicode character encoding set. The string data type MUST be encoded in UTF-8 [RFC3629] format. The string is sent as an array of zero or more octets using an Information Element of fixed or variable length. IPFIX Exporting Processes MUST NOT send IPFIX Messages containing ill-formed UTF-8 string values for Information Elements of the string data type; Collecting Processes SHOULD detect and ignore such values. See [UTF8-EXPLOIT] for background on this issue.

The data type octetArray has no encoding rules; it represents a raw array of zero or more octets, with the interpretation of the octets defined in the Information Element definition.

6.1.7. dateTimeSeconds

The data type dateTimeSeconds is an unsigned 32-bit integer in network byte order containing the number of seconds since the UNIX epoch, 1 January 1970 at 00:00 UTC, as defined in [POSIX.1]. dateTimeSeconds is encoded identically to the IPFIX Message Header Export Time field. It can represent dates between 1 January 1970 and 7 February 2106 without wraparound; see section 5.2 for wraparound considerations.

6.1.8. dateTimeMilliseconds

The data type dateTimeMilliseconds is an unsigned 64-bit integer in network byte order, containing the number of milliseconds since the UNIX epoch, 1 January 1970 at 00:00 UTC, as defined in [POSIX.1]. It can represent dates beginning on 1 January 1970 for approximately the next 500 billion years without wraparound.

6.1.9 dateTimeMicroseconds

The data type dateTimeMicroseconds is a 64-bit field encoded according to the NTP Timestamp format as defined in section 6 of [RFC5905]. This field is made up of two unsigned 32-bit integers in network byte order, Seconds and Fraction. The Seconds field is the number of seconds since the NTP epoch, 1 January 1900 at 00:00 UTC. The Fraction field is the fractional number of seconds in units of 1/(2^32) seconds (approximately 233 picoseconds). It can represent dates beginning between 1 January 1900 and 8 February 2036 in the current NTP Era; see section 5.2 for wraparound considerations.

Note that dateTimeMicroseconds and dateTimeNanoseconds share an identical encoding. The dateTimeMicroseconds data type is intended only to represent timestamps of microsecond precision. Therefore, the bottom 11 bits of the fraction field SHOULD be zero and MUST be ignored for all Information Elements of this data type (as 2^11 x 233 picoseconds = .477 microseconds).

6.1.10 dateTimeNanoseconds

The data type dateTimeNanoseconds is a 64-bit field encoded according to the NTP Timestamp format as defined in section 6 of [RFC5905]. This field is made up of two unsigned 32-bit integers in network byte order, Seconds and Fraction. The Seconds field is the number of seconds since the NTP epoch, 1 January 1900 at 00:00 UTC. The

Fraction field is the fractional number of seconds in units of 1/(2^32) seconds (approximately 233 picoseconds). It can represent dates beginning between 1 January 1900 and 8 February 2036 in the current NTP Era; see section 5.2 for wraparound considerations.

Note that dateTimeMicroseconds and dateTimeNanoseconds share an identical encoding. There is no restriction on the interpretation of the Fraction field for the dateTimeNanoseconds data type.

6.2. Reduced Size Encoding

Information Elements encoded as signed, unsigned, or float data types MAY be encoded using fewer octets than those implied by their type in the information model definition, based on the assumption that the smaller size is sufficient to carry any value the Exporter may need to deliver. This reduces the network bandwidth requirement between the Exporter and the Collector. Note that the Information Element definitions [IPFIX-IANA] always define the maximum encoding size.

For instance, the information model defines octetDeltaCount as an unsigned64 type, which would require 64 bits. However, if the Exporter will never locally encounter the need to send a value larger than 4294967295, it may chose to send the value instead as an unsigned32.

This behavior is indicated by the Exporter by specifying a size in the Template with a smaller length than that associated with the assigned type of the Information Element. In the example above, the Exporter would place a length of 4 versus 8 in the Template.

Reduced size encoding MAY be be applied to the following integer types: unsigned64, signed64, unsigned32, signed32, unsigned16, and signed16. The signed versus unsigned property of the reported value MUST be preserved. The reduction in size can be to any number of octets smaller than the original type if the data value still fits, i.e., so that only leading zeroes are dropped. For example, an unsigned64 can be reduced in size to 7, 6, 5, 4, 3, 2, or 1 octet(s).

Reduced size encoding MAY be used to reduce float64 to float32. The float32 not only has a reduced number range, but due to the smaller mantissa, is also less precise. In this case, the float64 would be reduced in size to 4 octets.

Reduced size encoding MUST NOT be applied to any other data type defined in [RFC5102bis] that implies a fixed length, as these types either have internal structure (such as ipv4Address or dateTimeMicroseconds) or restricted ranges that are not suitable for reduced length encoding (such as dateTimeMilliseconds).

Information Elements of type octetArray and string may be exported using any length, subject to restrictions on length specific to each Information Element, as noted in that Information Element's description.

7. Variable-Length Information Element

The IPFIX Template mechanism is optimized for fixed-length Information Elements [RFC5102bis]. Where an Information Element has a variable length, the following mechanism MUST be used to carry the length information for both the IANA and enterprise-specific Information Elements.

In the Template Set, the Information Element Field Length is recorded as 65535. This reserved length value notifies the Collecting Process that length of the Information Element will be carried in the Information Element content itself.

In most cases, the length of the Information Element will be less than 255 octets. The following length-encoding mechanism optimizes the overhead of carrying the Information Element length in this majority case. The length is carried in the octet before the Information Element, as shown in Figure R.

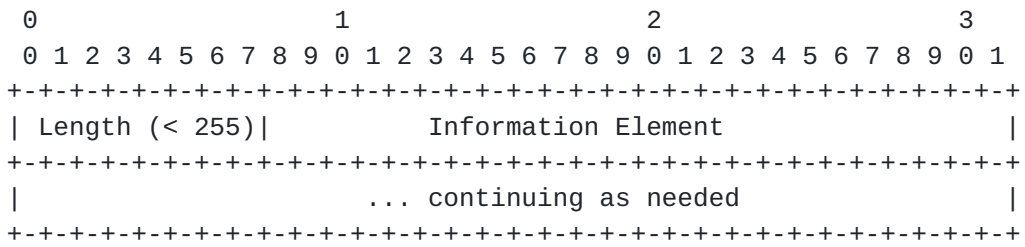


Figure R: Variable-Length Information Element (length < 255 octets)

The length may also be encoded into 3 octets before the Information element allowing the length of the Information Element to be greater than or equal to 255 octets. In this case, first octet of the Length field MUST be 255, and the length is carried in the second and third octets, as shown in Figure S.

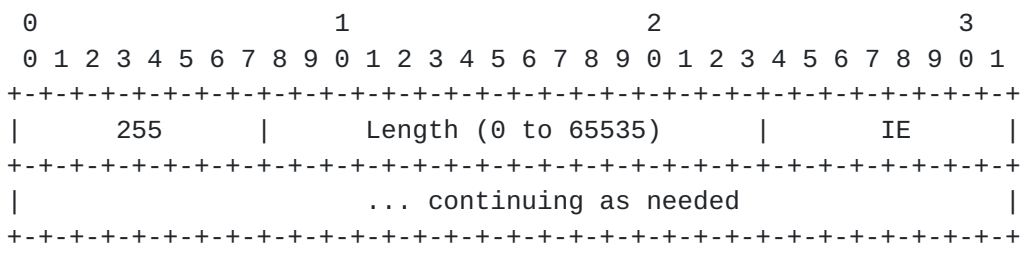


Figure S: Variable-Length Information Element (length 0 to 65535 octets)

The octets carrying the length (either the first or the first three octets) MUST NOT be included in the length of the Information Element.

8. Template Management

This section describes the management of Templates and Options Templates at the Exporting and Collecting Processes. The goal of Template management is to ensure, to the extent possible, that the Exporting Process and Collecting Process have a consistent view of the Templates and Options Templates used to encode and decode the Records sent from the Exporting Process to the Collecting Process. Achieving this goal is complicated somewhat by two factors: 1. the need to support the reuse of Template IDs within a Transport Session and 2. the need to support unreliable transmission for Templates when UDP is used as the transport protocol for IPFIX Messages.

The Template Management mechanisms defined in this section apply to IPFIX Messages export on SCTP, TCP, or UDP. Additional considerations specific to SCTP and UDP transport are given in sections 8.3 and 8.4, respectively.

The Exporting Process assigns and maintains Template IDs per Transport Session and Observation Domain. A newly created Template Record is assigned an unused Template ID by the Exporting Process. The Collecting Process MUST store all received Template Record information for the duration of each Transport Session until reuse or withdrawal as in section 8.1, or expiry over UDP as in section 8.4, so that it can interpret the corresponding Data Records.

The Collecting Process MUST NOT assume that the Template IDs from a given Exporting Process refer to the same Templates as they did in previous Transport Sessions from the same Exporting Process; a Collecting Process MUST NOT use Templates from one Transport Session to decode Data Sets in a subsequent Transport Session.

If a specific Information Element is required by a Template, but is not present in observed packets, the Exporting Process MAY choose to export Flow Records without this Information Element in a Data Record described by a new Template.

If an Information Element is required more than once in a Template, the different occurrences of this Information Element SHOULD follow the logical order of their treatments by the Metering Process. For example, if a selected packet goes through two hash functions, and if the two hash values are sent within a single Template, the first occurrence of the hash value should belong to the first hash function in the Metering Process. For example, when exporting the two source IP addresses of an IPv4-in-IPv4 packet, the first sourceIPv4Address Information Element occurrence should be the IPv4 address of the outer header, while the second occurrence should be the address of the inner header. Collecting Processes MUST properly handle Templates with multiple identical Information Elements.

The Exporting Process SHOULD transmit the Template Set and Options Template Set in advance of any Data Sets that use that (Options) Template ID, to help ensure that the Collector has the Template Record before receiving the first Data Record. Data Records that correspond to a Template Record MAY appear in the same and/or subsequent IPFIX Message(s). However, a Collecting Process MUST NOT assume that the Data Set and the associated Template Set (or Options Template Set) are exported in the same IPFIX Message.

Though a Collecting Process normally receives Template Records from the Exporting Process before receiving Data Records, this is not always the case, e.g. in case of reordering or Collecting Process restart over UDP. In these cases, the Collecting Process MAY buffer Data Records for which it has no Templates to wait for Template Records describing them; however, note that in the presence of Template withdrawal and redefinition (Section 8.1) this may lead to incorrect interpretation of Data Records.

Different Observation Domains within a Transport Session MAY use the same Template ID value to refer to different Templates; Collecting Processes MUST properly handle this case.

Options Templates and Templates which are related or interdependent (e.g. by sharing common properties as in [RFC5473]) SHOULD be sent together in the same IPFIX Message.

8.1. Template Withdrawal and Redefinition

Templates that will not be used further by an Exporting Process MAY be withdrawn by sending a Template Withdrawal. After receiving a

Template Withdrawal, a Collecting Process MUST stop using the Template to interpret subsequently-exported Data Sets. Note that this mechanism does not apply when UDP is used to transport IPFIX Messages; for this case, see Section 8.4.

A Template Withdrawal consists of a Template Record for the Template ID to be withdrawn, with a Field Count of 0. The format of a Template Withdrawal is shown in Figure T.

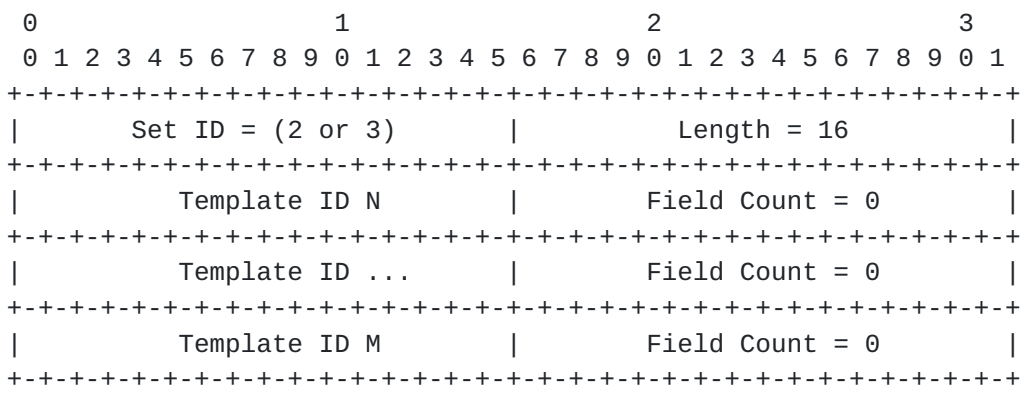


Figure T: Template Withdrawal Format

The Set ID field MUST contain the value 2 for Template Set Withdrawal and the value 3 for Options Template Set Withdrawal. Multiple Template IDs MAY be withdrawn with a single Template Withdrawal, in that case, padding MAY be used.

Template Withdrawals MAY appear interleaved with Template Sets, Options Template Sets, and Data Sets within an IPFIX Message. In this case, the Templates and Template Withdrawals shall be taken to take effect in the order in which they appear in the IPFIX Message. An Exporting Process SHOULD NOT send a Template Withdrawal until sufficient time has elapsed to allow receipt and processing of any Data Records described by the withdrawn Templates; see Section 8.2 on sequencing of Template management actions.

The end of a Transport Session implicitly withdraws all the Templates used within the Transport Session, and Templates must be resent during subsequent Transport Sessions between an Exporting Process and Collecting Process. This applies to SCTP and TCP only; see sections 8.4 and 10.3.4 for a discussion of Transport Session and Template lifetime over UDP.

All Templates for a given Observation Domain MAY also be withdrawn using an All Templates Withdrawal, shown in Figure U. All Options Templates for a given Observation Domain MAY likewise be withdrawn

using an All Options Templates Withdrawal, shown in Figure 3.

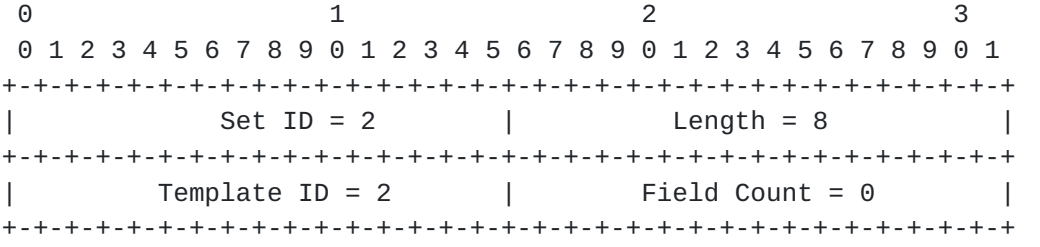


Figure U: All Templates Withdrawal Set Format

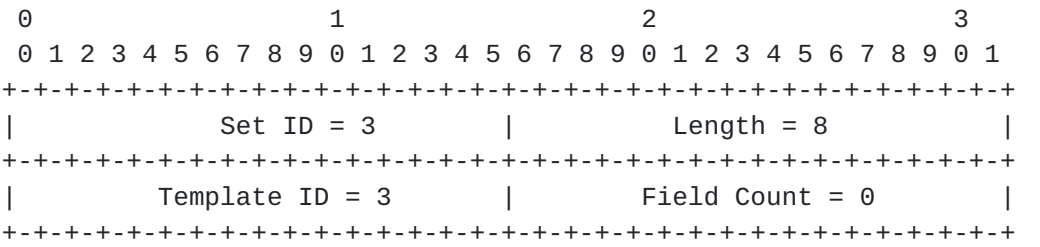


Figure V: All Options Templates Withdrawal Set Format

Template IDs MAY be reused for new Templates by sending a new Template Record or Options Template Record for a given Template ID after withdrawing the Template.

If a Collecting Process receives a Template Withdrawal for a Template or Options Template it does not presently have stored, this indicates a malfunctioning or improperly-implemented Exporting Process. The continued receipt and interpretation of Data Records is still possible, but it MUST ignore the Template Withdrawal and SHOULD log the error.

If a Collecting Process receives a new Template Record or Options Template Record for an already-allocated Template ID, and that Template or Options Template is identical to the already-received Template or Options Template, it SHOULD log the retransmission; however, this is not an error condition, as it does not affect the interpretation of data records.

If a Collecting Process receives a new Template Record or Options Template Record for an already-allocated Template ID, and that Template or Options Template is different from the already-received Template or Options Template, this indicates a malfunctioning or improperly-implemented Exporting Process. The continued receipt and unambiguous interpretation of Data Records for this Template ID is no longer possible, the Collecting Process SHOULD log the error; further Collecting Process actions are out of scope of this specification.

8.2 Sequencing Template Management Actions

Since there is no guarantee of the ordering of exported IPFIX Messages across SCTP Streams or over UDP, an Exporting Process MUST sequence all Template management actions (i.e., Template Records defining new Templates and Template Withdrawals withdrawing them) using the Export Time field in the IPFIX Message Header.

An Exporting Process MUST NOT export a Data Set described by a new Template in an IPFIX Message with an Export Time before the Export Time of the IPFIX Message containing that Template. If a new Template and a Data Set described by it appear in the same IPFIX Message, the Template Set containing the Template MUST appear before the Data Set in the Message.

An Exporting Process MUST NOT export any Data Sets described by a withdrawn Template in IPFIX Messages with an Export Time after the Export Time of the IPFIX Message containing the Template Withdrawal withdrawing that Template.

Put another way, a Template describes Data Records contained in IPFIX Messages with an Export Time between the Export Time of the IPFIX Message containing the Template Record and either the Export Time of the IPFIX Message containing the Template Withdrawal withdrawing it or the end of the Transport Session, inclusive.

Even if sent in-order, IPFIX Messages containing Template management actions could arrive at the Collecting Process out-of-order, i.e. if sent via UDP or via different SCTP streams. Given this, Template Withdrawals and subsequent reuse of Template IDs can significantly complicate the problem of determining Template lifetimes at the Collecting Process. A Collecting Process MAY implement a buffer and use Export Time information to disambiguate the order of Template management actions. This buffer, if implemented, SHOULD be configurable to impart a delay on the order of the maximum reordering delay experienced at the Collecting Process. Note, in this case, that the Collecting Process' clock is irrelevant: it is only comparing the Export Times of Messages to each other.

8.3. Additional considerations for Template Management over SCTP

The specifications in this section apply only to SCTP; in case of contradiction with specifications in Sections 8 or 8.1, this section takes precedence.

Template Sets and Options Template Sets MAY be sent on any SCTP stream. Data Sets sent on a given SCTP stream MAY be represented by Template Records exported on any SCTP stream.

Template Sets and Options Template Sets MUST be sent reliably, using SCTP ordered delivery.

Template Withdrawals MAY be sent on any SCTP stream. Template Withdrawals MUST be sent reliably, using SCTP ordered delivery. Template IDs MAY be reused by sending a Template Withdrawal and/or a new Template Record on a different SCTP stream than the stream on which the original Template was sent.

Additional Template Management considerations are given in [RFC6526], which specifies an extension to explicitly link Templates with SCTP streams. In exchange for more restrictive rules on the assignment of Template Records to SCTP streams, this extension allows fast, reliable reuse of Template IDs and estimation of Data Record loss per Template.

8.4. Additional considerations for Template Management over UDP

The specifications in this section apply only to UDP; in case of contradiction with specifications in Sections 8 or 8.1, this section takes precedence.

Since UDP provides no method for reliable transmission of Templates, Exporting Processes using UDP as the Transport Protocol MUST periodically retransmit each active Template at regular intervals. The Template retransmission interval MUST be configurable, as via the the templateRefreshTimeout and optionsTemplateRefreshTimeout defined in [RFC6728]. Default settings for these values are deployment- and application-specific.

Before exporting any Data Records described by a given Template Record or Options Template Record, especially in the case of Template ID reuse as in section 8.1, the Exporting Process SHOULD send multiple copies of the Template Record in separate IPFIX Message, in order to help ensure the Collecting Process has received it.

In order to minimize resource requirements for Templates which are no longer being used by the Exporting Process, the Collecting Process MAY associate a lifetime with each Template received in a UDP Transport Session. Templates not refreshed by the Exporting Process within the lifetime can then be discarded by the Collecting Process. The Template lifetime at the Collecting Process MAY be exposed by a configuration parameter, or MAY be derived from observation of the interval of periodic Template retransmissions from the Exporting Process. In this latter case, the Template lifetime SHOULD default to at least 3 times the observed retransmission rate.

Template Withdrawals (Section 8.1) MUST NOT be sent by Exporting

Processes exporting via UDP, and MUST be ignored by Collecting Processes collecting via UDP. Template IDs MAY be reused by Exporting Processes by exporting a new Template for the Template ID after waiting at least 3 times the retransmission rate. Note that Template ID reuse may lead to incorrect interpretation of Data Records if the retransmission and lifetime are not properly configured.

When a Collecting Process receives a new Template Record or Options Template Record via UDP for an already-allocated Template ID, and that Template or Options Template is identical to the already-received Template or Options Template, it SHOULD NOT log the retransmission, as this is the normal operation of Template refresh over UDP.

When a Collecting Process receives a new Template Record or Options Template Record for an already-allocated Template ID, and that Template or Options Template is different from the already-received Template or Options Template, the Collecting Process MUST replace the Template or Options Template for that Template ID with the newly-received Template or Options Template. This is the normal operation of Template ID reuse over UDP.

As Template IDs are unique per UDP session and per Observation Domain, at any given time, the Collecting Process SHOULD maintain the following for all the current Template Records and Options Template Records: <IPFIX Device, Exporter source UDP port, Collector IP address, Collector destination UDP port, Observation Domain ID, Template ID, Template Definition, Last Received>.

9. The Collecting Process's Side

This section describes the handling of the IPFIX Protocol at the Collecting Process common to all Transport Protocols. Additional considerations for SCTP and UDP are given in Sections 9.1 and 9.2 respectively. Template management at Collecting Processes is covered in Section 8.

The Collecting Process MUST listen for association requests / connections to start new Transport Sessions from the Exporting Process.

The Collecting Process MUST note the Information Element identifier of any Information Element that it does not understand and MAY discard that Information Element from received Data Records.

The Collecting Process MUST accept padding in Data Records and Template Records. The padding size is the Set Length minus the size of the Set Header (4 octets for the Set ID and the Set Length),

modulo the Record size deduced from the Template Record.

The IPFIX protocol has a Sequence Number field in the Export header that increases with the number of IPFIX Data Records in the IPFIX Message. A Collector can detect out-of-sequence, dropped, or duplicate IPFIX Messages by tracking the Sequence Number. A Collector SHOULD provide a logging mechanism for tracking out-of-sequence IPFIX Messages. Such out-of-sequence IPFIX Messages may be due to Exporter resource exhaustion where it cannot transmit messages at their creation rate, an Exporting Process reset, congestion on the network link between the Exporter and Collector, Collector resource exhaustion where it cannot process the IPFIX Messages at their arrival rate, out-of-order packet reception, duplicate packet reception, or an attacker injecting false messages.

If the Collecting Process receives a malformed IPFIX Message it MUST discard the IPFIX Message and SHOULD log the error. A malformed IPFIX Message is one that cannot be interpreted due to nonsensical length values (e.g., a variable length Information Element longer than its enclosing Set, a Set longer than its enclosing IPFIX Message, an IPFIX Message shorter than an IPFIX Message Header) or a reserved Version value (which may indicate a future version of IPFIX is being used for export, but practically occurs most often when non-IPFIX data is sent to an IPFIX Collecting Process). Note that non-zero Set padding does not constitute a malformed IPFIX Message.

9.1. Additional considerations for SCTP Collecting Processes

As an Exporting Process may request and support more than one stream per SCTP association, the Collecting Process MUST support the opening of multiple SCTP streams.

9.2. Additional considerations for UDP Collecting Processes

A Transport Session for IPFIX Messages transported over UDP is defined from the point of view of the Exporting Process, and roughly corresponds to the time during which a given Exporting Process sends IPFIX Messages over UDP to a given Collecting Process. Since this is difficult to detect at the Collecting Process, the Collecting Process MAY discard all Transport Session state after no IPFIX Messages are received from a given Exporting Process within a given Transport Session during a configurable idle timeout.

The Collecting Process SHOULD accept Data Records without the associated Template Record (or other definitions such as Common Properties) required to decode the Data Record. If the Template Records or other definitions have not been received at the time Data Records are received, the Collecting Process MAY store the Data

Records for a short period of time and decode them after the Template Records or other definitions are received, comparing Export Times of IPFIX Messages containing the Template Records with those containing the Data Records as in Section 8.2. Note that this mechanism may lead to incorrectly interpreted records in the presence of Template ID reuse or other identifiers with limited lifetimes.

10. Transport Protocol

The IPFIX Protocol Specification has been designed to be transport protocol independent. Note that the Exporter can export to multiple Collecting Processes using independent transport protocols.

The IPFIX Message Header 16-bit Length field limits the length of an IPFIX Message to 65535 octets, including the header. A Collecting Process MUST be able to handle IPFIX Message lengths of up to 65535 octets.

10.1. Transport Compliance and Transport Usage

SCTP [RFC4960] using the PR-SCTP extension specified in [RFC3758] MUST be implemented by all compliant implementations. UDP [UDP] MAY also be implemented by compliant implementations. TCP [TCP] MAY also be implemented by compliant implementations.

SCTP should be used in deployments where Exporters and Collectors are communicating over links that are susceptible to congestion. SCTP is capable of providing any required degree of reliability when used with the PR-SCTP extension.

TCP may be used in deployments where Exporters and Collectors communicate over links that are susceptible to congestion, but SCTP is preferred due to its ability to limit back pressure on Exporters and its message versus stream orientation.

UDP may be used, although it is not a congestion-aware protocol. However, in this case the IPFIX traffic between Exporter and Collector must be separately contained or provisioned to minimize the risk of congestion-related loss.

By default, the Collecting Process listens for connections on SCTP, TCP, and/or UDP port 4739. By default, the Collecting Process listens for secure connections on SCTP, TCP, and/or UDP port 4740 (refer to the Security Considerations section). By default, the Exporting Process attempts to connect to one of these ports. It MUST be possible to configure both the Exporting and Collecting Processes to use different ports than the default.

10.2. SCTP

This section describes how IPFIX is transported over SCTP [RFC4960] using the PR-SCTP [RFC3758] extension.

10.2.1. Congestion Avoidance

The SCTP transport protocol provides the required level of congestion avoidance by design.

SCTP detects congestion in the end-to-end path between the IPFIX Exporting Process and the IPFIX Collecting Process, and limits the transfer rate accordingly. When an IPFIX Exporting Process has records to export, but detects that transmission by SCTP is temporarily impossible, it can either wait until sending is possible again, or it can decide to drop the record. In the latter case, the dropped export data SHOULD be accounted for, so that the amount of dropped export data can be reported using the mechanism in Section 4.3.

10.2.2. Reliability

The SCTP transport protocol is by default reliable, but has the capability to deliver messages with partial reliability [RFC3758].

Using reliable SCTP messages for the IPFIX export is not in itself a guarantee that all Data Records will be delivered. If there is congestion on the link from the Exporting Process to the Collecting Process, or if a significant number of retransmissions are required, the send queues on the Exporting Process may fill up; the Exporting Process MAY either suspend, export, or discard the IPFIX Messages. If Data Records are discarded the IPFIX Sequence Numbers used for export MUST reflect the loss of data.

10.2.3. MTU

SCTP provides the required IPFIX Message fragmentation service based on path MTU discovery.

10.2.4. Association Establishment and Shutdown

The IPFIX Exporting Process initiates an SCTP association with the IPFIX Collecting Process. The Exporting Process MAY establish more than one association (connection "bundle" in SCTP terminology) to the Collecting Process.

An Exporting Process MAY support more than one active association to different Collecting Processes (including the case of different Collecting Processes on the same host).

When an Exporting Process is shut down, it SHOULD shut down the SCTP association.

When a Collecting Process no longer wants to receive IPFIX Messages, it SHOULD shut down its end of the association. The Collecting Process SHOULD continue to receive and process IPFIX Messages until the Exporting Process has closed its end of the association.

When a Collecting Process detects that the SCTP association has been abnormally terminated, it MUST continue to listen for a new association establishment.

When an Exporting Process detects that the SCTP association to the Collecting Process is abnormally terminated, it SHOULD try to re-establish the association.

Association timeouts SHOULD be configurable.

10.2.5. Failover

If the Collecting Process does not acknowledge an attempt by the Exporting Process to establish an association, SCTP will automatically retry association establishment using exponential backoff. The Exporter MAY log an alarm if the underlying SCTP association establishment times out; this timeout should be configurable on the Exporter.

The Exporting Process MAY open a backup SCTP association to a Collecting Process in advance, if it supports Collecting Process failover.

10.2.6. Streams

An Exporting Process MAY request more than one SCTP stream per association. Each of these streams may be used for the transmission of IPFIX Messages containing Data Sets, Template Sets, and/or Options Template Sets.

Depending on the requirements of the application, the Exporting Process may send Data Sets with full or partial reliability, using ordered or out-of-order delivery, over any SCTP stream established during SCTP Association setup.

An IPFIX Exporting Process MAY use any PR-SCTP Service Definition as per Section 4 of the PR-SCTP [RFC3758] specification when using partial reliability to transmit IPFIX Messages containing only Data Sets.

However, Exporting Processes SHOULD mark such IPFIX Messages for retransmission for as long as resource or other constraints allow.

10.3. UDP

This section describes how IPFIX is transported over UDP [UDP].

10.3.1. Congestion Avoidance

UDP has no integral congestion-avoidance mechanism. Its use over congestion-sensitive network paths is therefore not recommended. UDP MAY be used in deployments where Exporters and Collectors always communicate over dedicated links that are not susceptible to congestion, i.e., links that are over-provisioned compared to the maximum export rate from the Exporters.

10.3.2. Reliability

UDP is not a reliable transport protocol, and cannot guarantee delivery of messages. IPFIX Messages sent from the Exporting Process to the Collecting Process using UDP may therefore be lost. UDP MUST NOT be used unless the application can tolerate some loss of IPFIX Messages.

The Collecting Process SHOULD deduce the loss and reordering of IPFIX Data Records by looking at the discontinuities in the IPFIX Sequence Number. In the case of UDP, the IPFIX Sequence Number contains the total number of IPFIX Data Records sent for the UDP Transport Session prior to the receipt of this IPFIX Message, modulo 2^32. A Collector SHOULD detect out-of-sequence, dropped, or duplicate IPFIX Messages by tracking the Sequence Number.

Exporting Processes exporting IPFIX Messages via UDP MUST include a valid UDP checksum [UDP] in UDP datagrams including IPFIX messages.

10.3.3. MTU

The maximum size of exported messages MUST be configured such that the total packet size does not exceed the path MTU. If the path MTU is unknown, a maximum packet size of 512 octets SHOULD be used.

10.3.4. Session Establishment and Shutdown

As UDP is a connectionless protocol, there is no real session establishment or shutdown for IPFIX over UDP. An Exporting Process starts sending IPFIX Messages to a Collecting Process at one point in time, and stops sending them at another point in time. This can lead to some complications in Template management, which are outlined in Section 8.4 above.

10.3.5. Failover and Session Duplication

Because UDP is not a connection-oriented protocol, the Exporting Process is unable to determine from the transport protocol that the Collecting Process is no longer able to receive the IPFIX Messages. Therefore, it cannot invoke a failover mechanism. However, the Exporting Process MAY duplicate the IPFIX Message to several Collecting Processes.

10.4. TCP

This section describes how IPFIX is transported over TCP [TCP].

10.4.1. Congestion Avoidance

TCP controls the rate at which data can be sent from the Exporting Process to the Collecting Process, using a mechanism that takes into account both congestion in the network and the capabilities of the receiver.

Therefore, an IPFIX Exporting Process may not be able to send IPFIX Messages at the rate that the Metering Process generates them, either because of congestion in the network or because the Collecting Process cannot handle IPFIX Messages fast enough. As long as congestion is transient, the Exporting Process can buffer IPFIX Messages for transmission. But such buffering is necessarily limited, both because of resource limitations and because of timeliness requirements, so ongoing and/or severe congestion may lead to a situation where the Exporting Process is blocked.

When an Exporting Process has Data Records to export but the transmission buffer is full, and it wants to avoid blocking, it can decide to drop some Data Records. The dropped Data Records MUST be accounted for, so that the number of lost records can later be reported as in Section 4.3.

10.4.2. Reliability

TCP ensures reliable delivery of data from the Exporting Process to the Collecting Process.

10.4.3. MTU

As TCP offers a stream service instead of a datagram or sequential packet service, IPFIX Messages transported over TCP are instead separated using the Length field in the IPFIX Message Header. The Exporting Process can choose any valid length for exported IPFIX Messages, as TCP handles segmentation.

Exporting Processes may choose IPFIX Message lengths lower than the maximum in order to avoid head-of-line blocking and/or to ensure timely export of Data Records.

10.4.4. Connection Establishment and Shutdown

The IPFIX Exporting Process initiates a TCP connection to the Collecting Process. An Exporting Process MAY support more than one active connection to different Collecting Processes (including the case of different Collecting Processes on the same host). An Exporting Process MAY support more than one active connection to the same Collecting Process to avoid head of line blocking across Observation Domains.

The Exporter MAY log an alarm if the underlying TCP connection establishment times out; this timeout should be configurable on the Exporter.

When an Exporting Process is shut down, it SHOULD shut down the TCP connection.

When a Collecting Process no longer wants to receive IPFIX Messages, it SHOULD close its end of the connection. The Collecting Process SHOULD continue to read IPFIX Messages until the Exporting Process has closed its end.

When a Collecting Process detects that the TCP connection to the Exporting Process has terminated abnormally, it MUST continue to listen for a new connection.

When an Exporting Process detects that the TCP connection to the Collecting Process has terminated abnormally, it SHOULD try to re-establish the connection. Connection timeouts and retry schedules SHOULD be configurable. In the default configuration, an Exporting Process MUST NOT attempt to establish a connection more frequently than once per minute.

10.4.5. Failover

If the Collecting Process does not acknowledge an attempt by the Exporting Process to establish a connection, TCP will automatically retry connection establishment using exponential backoff. The Exporter MAY log an alarm if the underlying TCP connection establishment times out; this timeout should be configurable on the Exporter.

The Exporting Process MAY open a backup TCP connection to a Collecting Process in advance, if it supports Collecting Process failover.

11. Security Considerations

The security considerations for the IPFIX protocol have been derived from an analysis of potential security threats, as discussed in the "Security Considerations" section of IPFIX requirements [RFC3917]. The requirements for IPFIX security are as follows:

1. IPFIX must provide a mechanism to ensure the confidentiality of IPFIX data transferred from an Exporting Process to a Collecting Process, in order to prevent disclosure of Flow Records transported via IPFIX.

2. IPFIX must provide a mechanism to ensure the integrity of IPFIX data transferred from an Exporting Process to a Collecting Process, in order to prevent the injection of incorrect data or control information (e.g., Templates), or the duplication of Messages, in an IPFIX Message stream.

3. IPFIX must provide a mechanism to authenticate IPFIX Collecting and Exporting Processes, to prevent the collection of data from an unauthorized Exporting Process or the export of data to an unauthorized Collecting Process.

Because IPFIX can be used to collect information for network forensics and billing purposes, attacks designed to confuse, disable, or take information from an IPFIX collection system may be seen as a prime objective during a sophisticated network attack.

An attacker in a position to inject false messages into an IPFIX Message stream can either affect the application using IPFIX (by falsifying data), or the IPFIX Collecting Process itself (by modifying or revoking Templates, or changing options); for this reason, IPFIX Message integrity is important.

The IPFIX Messages themselves may also contain information of value to an attacker, including information about the configuration of the network as well as end-user traffic and payload data, so care must be taken to confine their visibility to authorized users. When an Information Element containing end-user payload information is exported, it SHOULD be transmitted to the Collecting Process using a means that secures its contents against eavesdropping. Suitable mechanisms include the use of either a direct point-to-point connection assumed to be unavailable to attackers, or the use of an encryption mechanism. It is the responsibility of the Collecting Process to provide a satisfactory degree of security for this collected data, including, if necessary, encryption and/or anonymization of any reported data; see Section 11.8.

11.1. Applicability of TLS and DTLS

Transport Layer Security (TLS) [RFC5246] and Datagram Transport Layer Security (DTLS) [RFC6347] were designed to provide the confidentiality, integrity, and authentication assurances required by the IPFIX protocol, without the need for pre-shared keys.

IPFIX Exporting Processes and Collecting Processes using TCP MUST support TLS version 1.1 and SHOULD support TLS version 1.2 [RFC5246], including the mandatory cipher suite(s) specified in each version. IPFIX Exporting Processes and Collecting Processes using UDP or SCTP MUST support DTLS 1.0 and SHOULD support DTLS version 1.2 [RFC6347], including the mandatory cipher suite(s) specified in each version.

Note that DTLS is selected as the security mechanism for SCTP. Though TLS bindings to SCTP are defined in [RFC3436], they require all communication to be over reliable, bidirectional streams, and require one TLS connection per stream. This arrangement is not compatible with the rationale behind the choice of SCTP as an IPFIX transport protocol.

Note that using DTLS has a vulnerability, i.e., a true man in the middle may attempt to take data out of an association and fool the sender into thinking that the data was actually received by the peer. In generic TLS for SCTP (and/or TCP), this is not possible. This means that the removal of a message may become hidden from the sender or receiver. Another vulnerability of using SCTP with DTLS is that someone could inject SCTP control information to shut down the SCTP

association, effectively generating a loss of IPFIX Messages if those are buffered outside of the SCTP association. Techniques such as [RFC6083] could be used to overcome these vulnerabilities.

When using DTLS over SCTP, the Exporting Process MUST ensure that each IPFIX Message is sent over the same SCTP stream that would be used when sending the same IPFIX Message directly over SCTP. Note that DTLS may send its own control messages on stream 0 with full reliability; however, this will not interfere with the processing of stream 0 IPFIX Messages at the Collecting Process, because DTLS consumes its own control messages before passing IPFIX Messages up to the application layer.

When using DTLS over SCTP or UDP, the Heartbeat Extension [RFC6520] SHOULD be used, especially on long-lived Transport Sessions, to ensure that the association remains active.

Exporting and Collecting Processes MUST NOT request, offer, or use any version of SSL, or any version of TLS prior to 1.1, due to known security vulnerabilities in prior versions of the protocol; see Appendix E of [RFC5246] for more information.

11.2. Usage

The IPFIX Exporting Process initiates the communication to the IPFIX Collecting Process, and acts as a TLS or DTLS client according to [RFC5246] and [RFC6347], while the IPFIX Collecting Process acts as a TLS or DTLS server. The DTLS client opens a secure connection on the SCTP port 4740 of the DTLS server if SCTP is selected as the transport protocol. The TLS client opens a secure connection on the TCP port 4740 of the TLS server if TCP is selected as the transport protocol. The DTLS client opens a secure connection on the UDP port 4740 of the DTLS server if UDP is selected as the transport protocol.

11.3. Mutual Authentication

When using TLS or DTLS, IPFIX Exporting Processes and IPFIX Collecting Processes SHOULD be identified by a certificate containing the DNS-ID identifier as in Section 6.4 of [RFC6125]; the inclusion of Common Names (CN-IDs) in certificates identifying IPFIX Exporting Processes or Collecting Processes is NOT RECOMMENDED.

To prevent man-in-the-middle attacks from impostor Exporting or Collecting Processes, the acceptance of data from an unauthorized Exporting Process, or the export of data to an unauthorized Collecting Process, mutual authentication MUST be used for both TLS and DTLS. Exporting Processes MUST verify the reference identifiers

of the Collecting Processes they are exporting IPFIX messages to against those stored in the certificates. Likewise, Collecting Processes MUST verify the reference identifiers of the Exporting Processes they are receiving IPFIX Messages from against those stored in the certificates. Exporting Processes MUST NOT export to non-verified Collecting Processes, and Collecting Processes MUST NOT accept IPFIX Messages from non-verified Exporting Processes.

Exporting Processes and Collecting Processes MUST support the verification of certificates against an explicitly authorized list of peer certificates identified by Common Name, and SHOULD support the verification of reference identifiers by matching the DNS-ID or CN-ID with a DNS lookup of the peer.

IPFIX Exporting Processes and Collecting Processes MUST use non-NULL ciphersuites for authentication, integrity, and confidentiality.

11.4. Protection against DoS Attacks

An attacker may mount a denial-of-service (DoS) attack against an IPFIX collection system either directly, by sending large amounts of traffic to a Collecting Process, or indirectly, by generating large amounts of traffic to be measured by a Metering Process.

Direct DoS attacks can also involve state exhaustion, whether at the transport layer (e.g., by creating a large number of pending connections), or within the IPFIX Collecting Process itself (e.g., by sending Flow Records pending Template or scope information, a large amount of Options Template Records, etc.).

SCTP mandates a cookie-exchange mechanism designed to defend against SCTP state exhaustion DoS attacks. Similarly, TCP provides the "SYN cookie" mechanism to mitigate state exhaustion; SYN cookies SHOULD be used by any Collecting Process accepting TCP connections. DTLS also provides cookie exchange to protect against DTLS server state exhaustion.

The reader should note that there is no way to prevent fake IPFIX Message processing (and state creation) for UDP & SCTP communication. The use of TLS and DTLS can obviously prevent the creation of fake states, but they are themselves prone to state exhaustion attacks. Therefore, Collector rate limiting SHOULD be used to protect TLS & DTLS (like limiting the number of new TLS or DTLS session per second to a sensible number).

IPFIX state exhaustion attacks can be mitigated by limiting the rate at which new connections or associations will be opened by the Collecting Process, the rate at which IPFIX Messages will be accepted

by the Collecting Process, and adaptively limiting the amount of state kept, particularly records waiting on Templates. These rate and state limits MAY be provided by a Collecting Process; if provided, the limits SHOULD be user configurable.

Additionally, an IPFIX Collecting Process can eliminate the risk of state exhaustion attacks from untrusted nodes by requiring TLS or DTLS mutual authentication, causing the Collecting Process to accept IPFIX Messages only from trusted sources.

With respect to indirect denial of service, the behavior of IPFIX under overload conditions depends on the transport protocol in use. For IPFIX over TCP, TCP congestion control would cause the flow of IPFIX Messages to back off and eventually stall, blinding the IPFIX system. SCTP improves upon this situation somewhat, as some IPFIX Messages would continue to be received by the Collecting Process due to the avoidance of head-of-line blocking by SCTP's multiple streams and partial reliability features, possibly affording some visibility of the attack. The situation is similar with UDP, as some datagrams may continue to be received at the Collecting Process, effectively applying sampling to the IPFIX Message stream, implying that some forensics may be left.

To minimize IPFIX Message loss under overload conditions, some mechanism for service differentiation could be used to prioritize IPFIX traffic over other traffic on the same link. Alternatively, IPFIX Messages can be transported over a dedicated network. In this case, care must be taken to ensure that the dedicated network can handle the expected peak IPFIX Message traffic.

11.5. When DTLS or TLS Is Not an Option

The use of DTLS or TLS might not be possible in some cases due to performance issues or other operational concerns.

Without TLS or DTLS mutual authentication, IPFIX Exporting Processes and Collecting Processes can fall back on using IP source addresses to authenticate their peers. A policy of allocating Exporting Process and Collecting Process IP addresses from specified address ranges, and using ingress filtering to prevent spoofing, can improve the usefulness of this approach. Again, completely segregating IPFIX traffic on a dedicated network, where possible, can improve security even further. In any case, the use of open Collecting Processes (those that will accept IPFIX Messages from any Exporting Process regardless of IP address or identity) is discouraged.

Modern TCP and SCTP implementations are resistant to blind insertion attacks (see [RFC4960], [RFC6528]); however, UDP offers no such

protection. For this reason, IPFIX Message traffic transported via UDP and not secured via DTLS SHOULD be protected via segregation to a dedicated network.

11.6. Logging an IPFIX Attack

IPFIX Collecting Processes MUST detect potential IPFIX Message insertion or loss conditions by tracking the IPFIX Sequence Number, and SHOULD provide a logging mechanism for reporting out-of-sequence messages. Note that an attacker may be able to exploit the handling of out-of-sequence messages at the Collecting Process, so care should be taken in handling these conditions. For example, a Collecting Process that simply resets the expected Sequence Number upon receipt of a later Sequence Number could be temporarily blinded by deliberate injection of later Sequence Numbers.

IPFIX Exporting and Collecting Processes SHOULD log any connection attempt that fails due to authentication failure, whether due to being presented an unauthorized or mismatched certificate during TLS or DTLS mutual authentication, or due to a connection attempt from an unauthorized IP address when TLS or DTLS is not in use.

IPFIX Exporting and Collecting Processes SHOULD detect and log any SCTP association reset or TCP connection reset.

11.7. Securing the Collector

The security of the Collector and its implementation is important to achieve overall security; however, a complete set of security guidelines for Collector implementation is outside the scope of this document.

As IPFIX uses length-prefix encodings, Collector implementors should take care to ensure detection of and proper operation despite inconsistent values that could impact IPFIX Message decoding. Specifically, IPFIX Message, Set, and variable-length Information Element lengths must be checked for consistency to avoid buffer-sizing vulnerabilities.

Collector implementors should also pay special attention to UTF-8 encoding of string datatypes, as vulnerabilities may exist in the interpretation of ill-formed UTF-8 values; see Section 6.1.6.

11.8. Privacy Considerations for Collected Data

Flow data exported by Exporting Processes, and collected by Collecting Processes, typically contains information about traffic on the observed network. This information may be personally identifiable

and privacy-sensitive. The storage of this data must be protected via technical as well as policy means to ensure that the privacy of the users of the measured network is protected. A complete specification of such means is out of scope for this document, and specific to the application and storage technology used.

12. IANA Considerations

On publication of this document, IANA will update the IPFIX Information Element Registry [IPFIX-IANA] to update all references to RFC5101 to point to this document, instead.

This document has no further actions for IANA; the text below is explanatory.

IPFIX Messages use two fields with assigned values. These are the IPFIX Version Number, indicating which version of the IPFIX Protocol was used to export an IPFIX Message, and the IPFIX Set ID, indicating the type for each set of information within an IPFIX Message.

The Information Elements used by IPFIX, and sub-registries of Information Element values, are managed by IANA [IPFIX-IANA], as are the Private Enterprise Numbers used by enterprise-specific Information Elements [PEN-IANA]. This document makes no changes to these registries.

The IPFIX Version Number value of 0x000a (10) is reserved for the IPFIX protocol specified in this document. Set ID values of 0 and 1 are not used, for historical reasons [RFC3954]. The Set ID value of 2 is reserved for the Template Set. The Set ID value of 3 is reserved for the Options Template Set. All other Set ID values from 4 to 255 are reserved for future use. Set ID values above 255 are used for Data Sets.

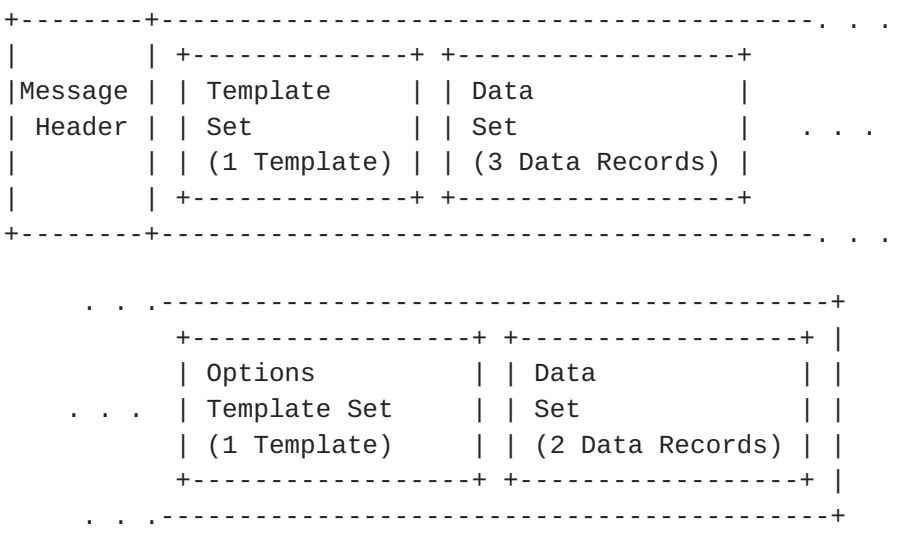
New assignments in either IPFIX Version Number or IPFIX Set ID assignments require a Standards Action [RFC5226], i.e., they are to be made via Standards Track RFCs approved by the IESG.

Appendix A. IPFIX Encoding Examples

This appendix, which is a not a normative reference, contains IPFIX encoding examples.

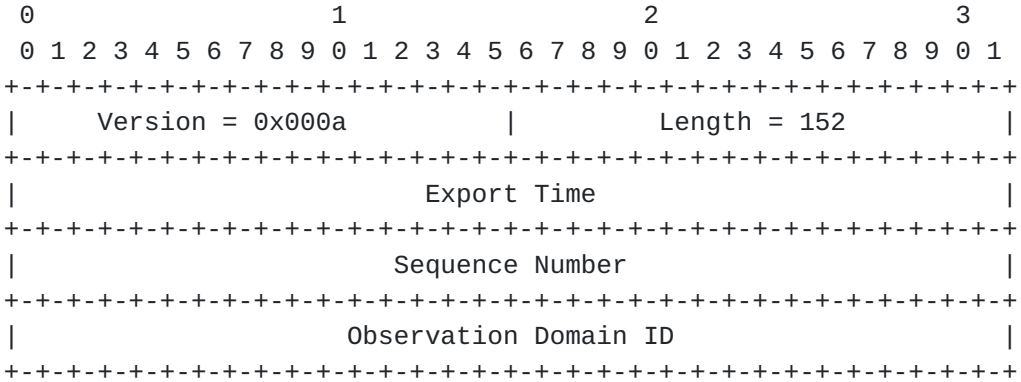
Let's consider the example of an IPFIX Message composed of a Template Set, a Data Set (which contains three Data Records), an Options Template Set and a Data Set (which contains 2 Data Records related to the previous Options Template Record).

IPFIX Message:



A.1. Message Header Example

The Message Header is composed of:



A.2. Template Set Examples

A.2.1. Template Set Using IANA Information Elements

We want to report the following Information Elements:

- The IPv4 source IP address: sourceIPv4Address in [IPFIX-IANA], with a length of 4 octets

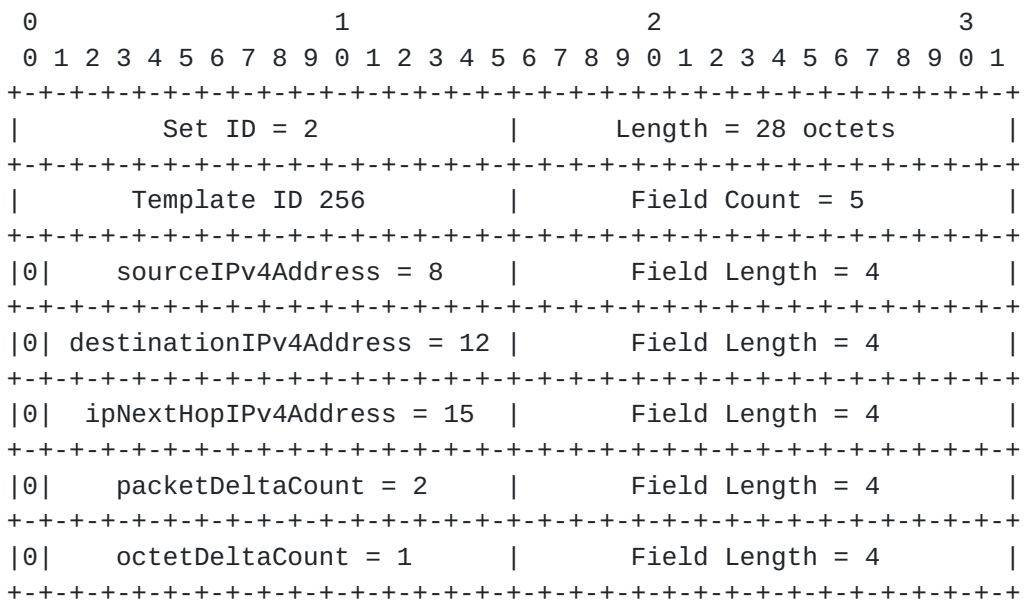
- The IPv4 destination IP address: destinationIPv4Address in [IPFIX-IANA], with a length of 4 octets

- The next-hop IP address (IPv4): ipNextHopIPv4Address in [IPFIX-IANA], with a length of 4 octets

- The number of packets of the Flow: packetDeltaCount in [IPFIX-IANA], with a length of 4 octets

- The number of octets of the Flow: octetDeltaCount in [IPFIX-IANA], with a length of 4 octets

Therefore, the Template Set will be composed of the following:



A.2.2. Template Set Using Enterprise-Specific Information Elements

We want to report the following Information Elements:

- The IPv4 source IP address: sourceIPv4Address in [IPFIX-IANA], with a length of 4 octets

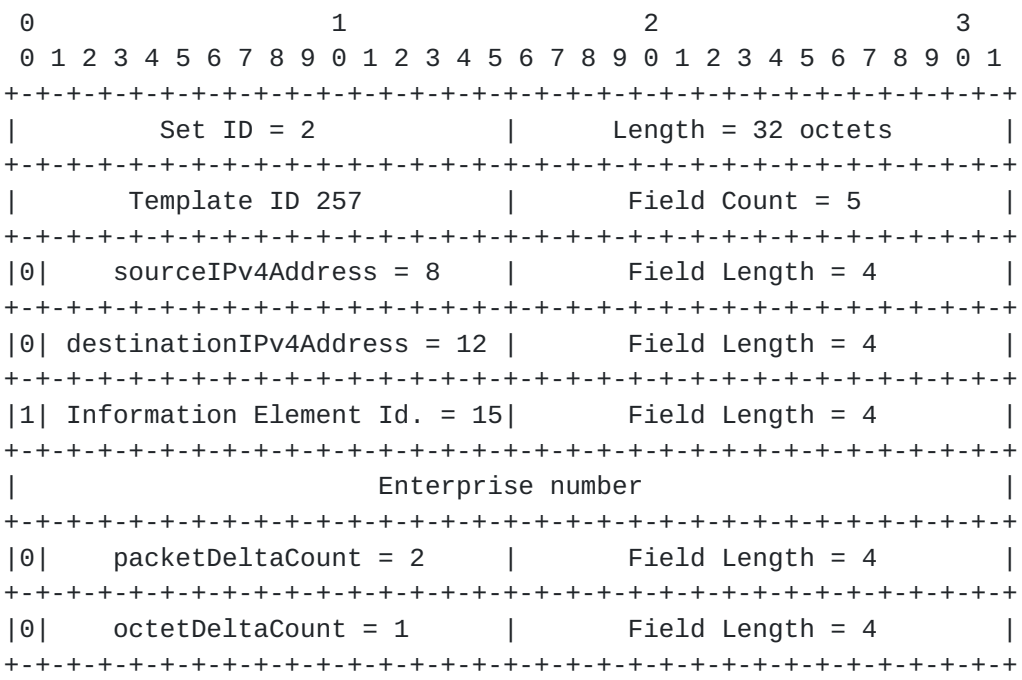
- The IPv4 destination IP address: destinationIPv4Address in [IPFIX-IANA], with a length of 4 octets

- An enterprise-specific Information Element representing proprietary information, with a type of 15 and a length of 4

- The number of packets of the Flow: packetDeltaCount in [IPFIX-IANA], with a length of 4 octets

- The number of octets of the Flow: octetDeltaCount in [IPFIX-IANA], with a length of 4 octets

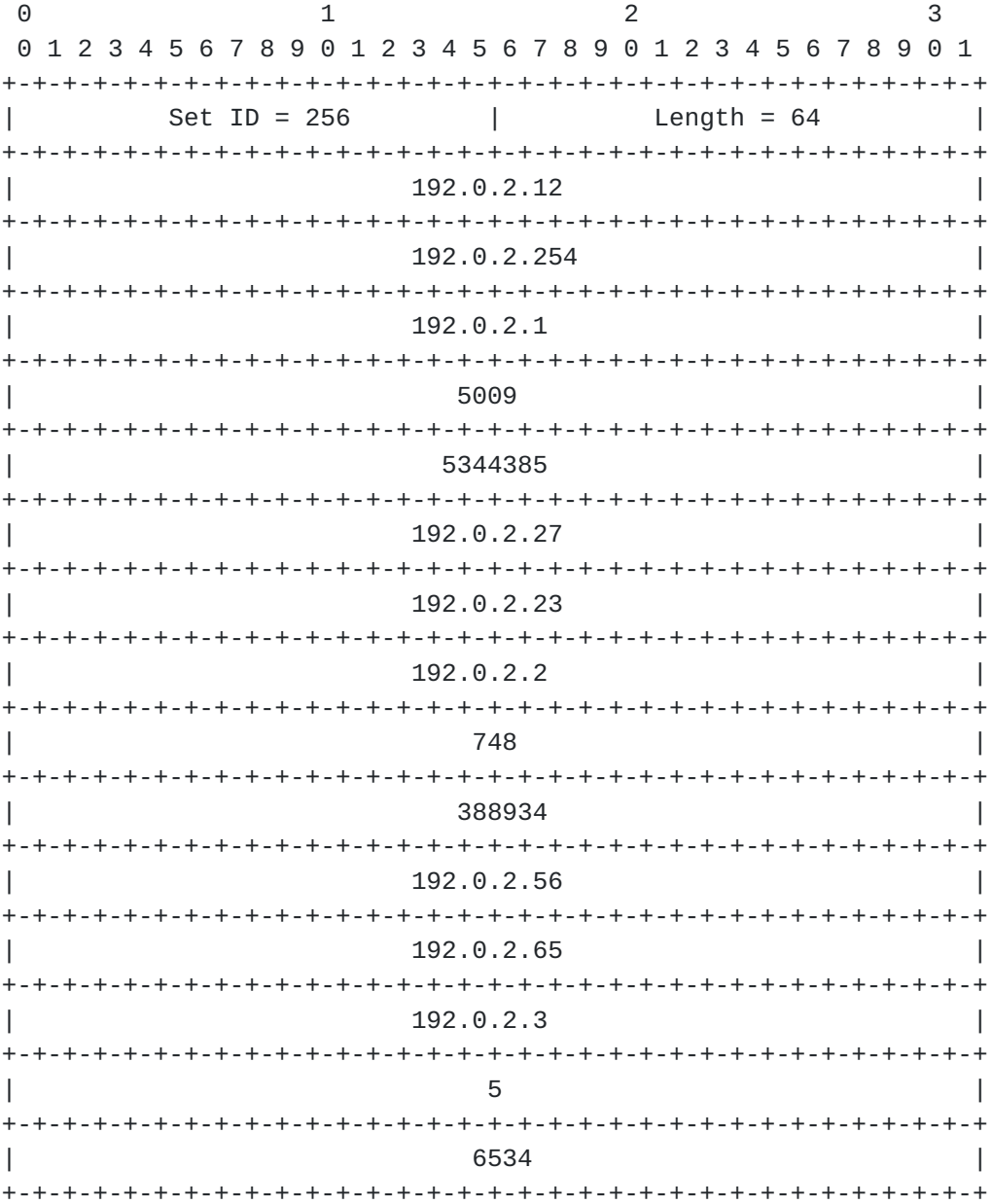
Therefore, the Template Set will be composed of the following:



A.3. Data Set Example

In this example, we report the following three Flow Records:

Src IP addr.	Dst IP addr.	Next Hop addr.	Packet Number	Octets Number
192.0.2.12	192.0.2.254	192.0.2.1	5009	5344385
192.0.2.27	192.0.2.23	192.0.2.2	748	388934
192.0.2.56	192.0.2.65	192.0.2.3	5	6534



Note that padding is not necessary in this example.

A.4. Options Template Set Examples

A.4.1. Options Template Set Using IANA Information Elements

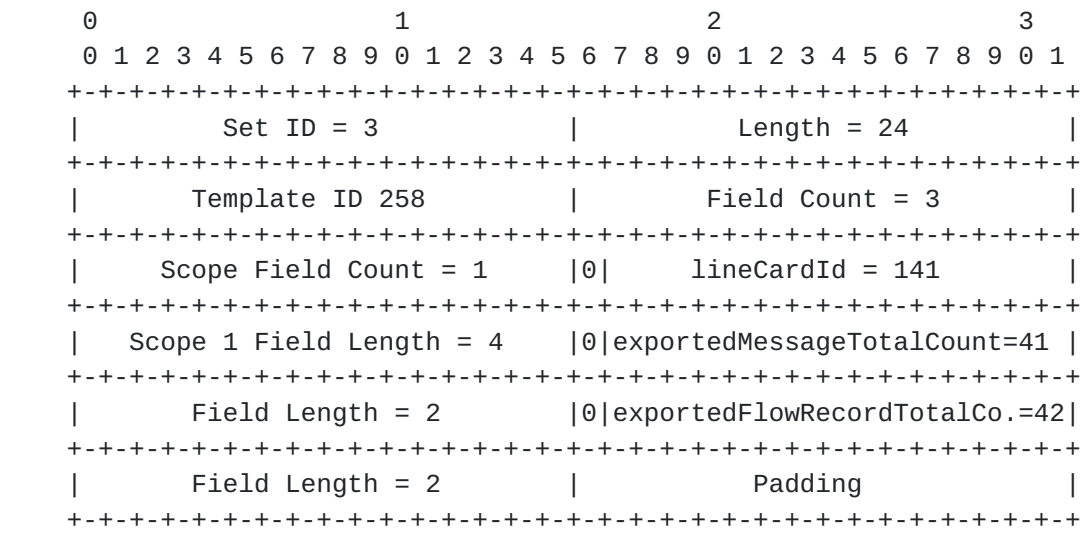
Per line card (the router being composed of two line cards), we want to report the following Information Elements:

- Total number of IPFIX Messages: exportedMessageTotalCount [IPFIX-IANA], with a length of 2 octets

- Total number of exported Flows: exportedFlowRecordTotalCount [IPFIX-IANA], with a length of 2 octets

The line card, which is represented by the lineCardId Information Element [IPFIX-IANA], is used as the Scope Field.

Therefore, the Options Template Set will be:



A.4.2. Options Template Set Using Enterprise-Specific Information Elements

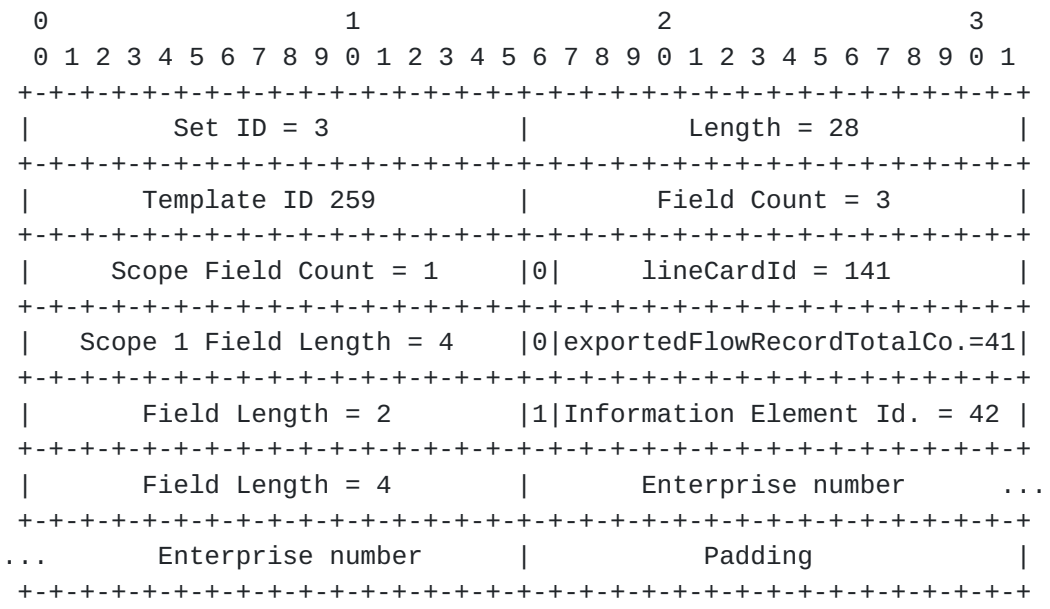
Per line card (the router being composed of two line cards), we want to report the following Information Elements:

- Total number of IPFIX Messages: exportedMessageTotalCount [IPFIX-IANA], with a length of 2 octets

- An enterprise-specific number of exported Flows, with a type of 42 and a length of 4 octets

The line card, which is represented by the lineCardId Information Element [IPFIX-IANA], is used as the Scope Field.

The format of the Options Template Set is as follows:



A.4.3. Options Template Set Using an Enterprise-Specific Scope

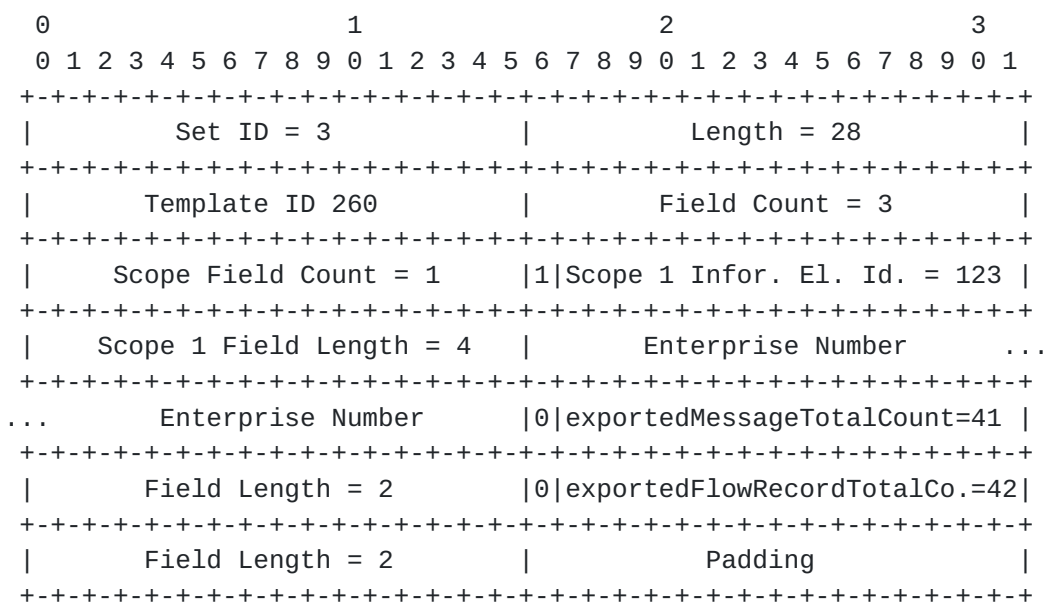
In this example, we want to export the same information as in the example in Section A.4.1:

- Total number of IPFIX Messages: exportedMessageTotalCount [IPFIX-IANA], with a length of 2 octets

- Total number of exported Flows: exportedFlowRecordTotalCount [IPFIX-IANA], with a length of 2 octets

But this time, the information pertains to a proprietary scope, identified by enterprise-specific Information Element number 123.

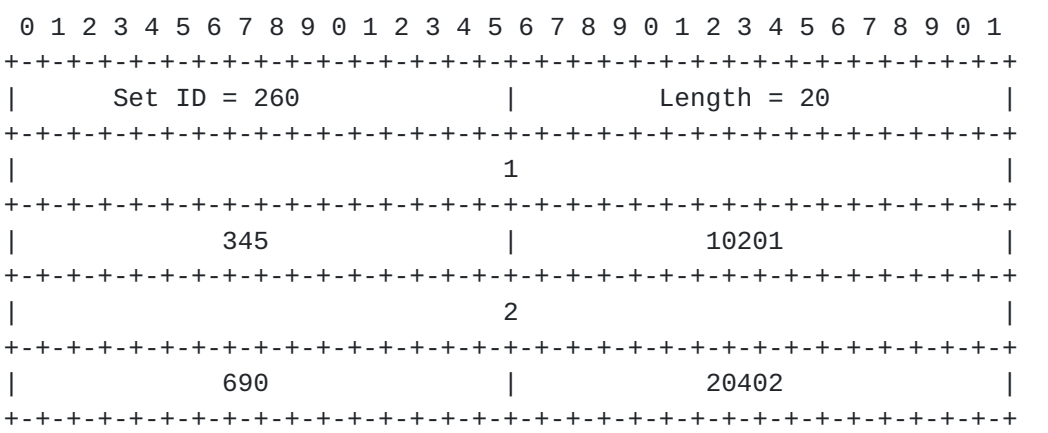
The format of the Options Template Set is now as follows:



A.4.4. Data Set Using an Enterprise-Specific Scope

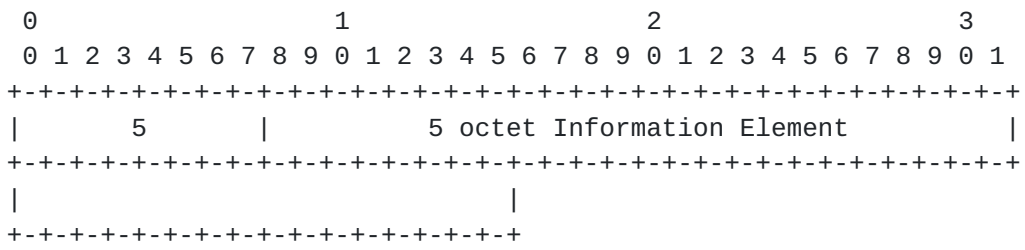
In this example, we report the following two Data Records:

Enterprise field 123	IPFIX Message	Exported Flow Records
1	345	10201
2	690	20402

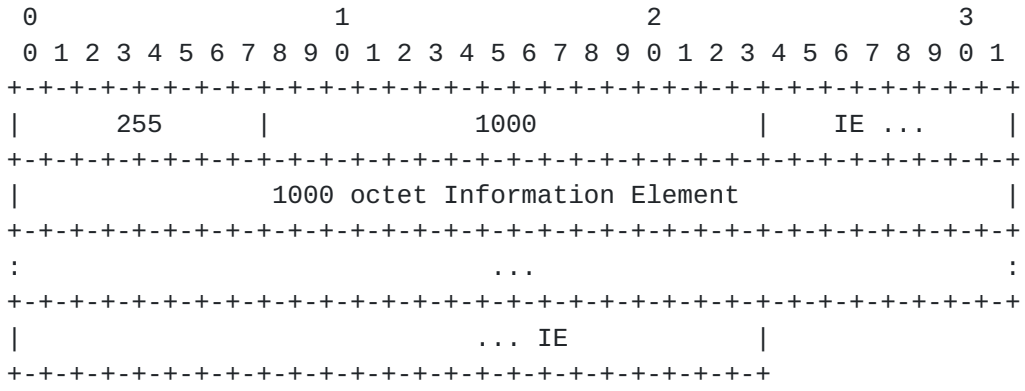


A.5. Variable-Length Information Element Examples

A.5.1. Example of Variable-Length Information Element with Length Inferior to 255 Octets



A.5.2. Example of Variable-Length Information Element with 3 Octet Length Encoding



References

Normative References

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

[RFC3436] Jungmaier, A., Rescorla, E., and M. Tuexen, "Transport Layer Security over Stream Control Transmission Protocol", RFC 3436, December 2002.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 3629, November 2003.

[RFC3758] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", RFC 3758, May 2004.

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

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

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

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

[RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, March 2011.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

[RFC6520] Seggelmann, R., Tuexen, M., and Williams, M., "Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension", RFC 6520, February 2012.

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

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

[RFC5102bis] Quittek, J., Bryant S., Claise, B., Aitken, P., and J. Meyer, "Information Model for IP Flow Information Export", draft-claise-ipfix-information-model-rfc5102bis-01.txt, Work in Progress, October 2011.

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

Informative References

[RFC2579] McCloghrie, K., Perkins, D., and J. Schoenwaelder, "Textual Conventions for SMIv2", STD 58, RFC 2579, April 1999.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

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

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

[RFC5101] Claise, B., Ed., "Bidirectional Flow Export Using IP Flow Information Export (IPFIX)", RFC 5103, January 2008.

[RFC5103] Trammell, B., and E. Boschi, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information", RFC 5101, January 2008.

[RFC5153] Boschi, E., Mark, L., Quittek J., and P. Aitken, "IP Flow Information Export (IPFIX) Implementation Guidelines", RFC5153, April 2008

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

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

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

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

[RFC5474] Duffield, N., Ed., Chiou, D., Claise, B., Greenberg, A., Grossglauser, M., and J. Rexford, "A Framework for Packet Selection and Reporting", RFC 5474, March 2009.

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

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

[RFC5610] Boschi, E., Trammell, B., Mark, L., and T. Zseby, "Exporting Type Information for IP Flow Information Export (IPFIX) Information Elements", RFC 5610, July 2009.

[RFC5655] Trammell, B., Boschi, E., Mark, L., Zseby, T., and A. Wagner, "Specification of the IP Flow Information Export (IPFIX) File Format", RFC 5655, October 2009.

[RFC6083] Tuexen, M., Seggelman, R. and E. Rescola, "Datagram Transport Layer Security (DTLS) for Stream Control Transmission Protocol (SCTP)", RFC6083, January 2011.

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

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

[RFC6526] Claise, B., Aitken, P., Johnson, A. and G. Muenz, "IPFIX Export per SCTP Stream", RFC 6526, March 2012.

[RFC6528] Gont, F. and S. Bellovin, "Defending Against Sequence Number Attacks", RFC 6528, February 2012.

[RFC6615] Dietz, T., Kobayashi, A., Claise, B., and G. Muenz, "Definitions of Managed Objects for IP Flow Information Export", RFC 6615, June 2012.

[RFC6727] Dietz, T. Ed., Claise, B., and J. Quittek, "Definitions of Managed Objects for Packet Sampling", RFC 6727, October 2012.

[RFC6728] Muenz, G., Claise, B., and P. Aitken, "Configuration Data Model for IPFIX and PSAMP", RFC 6728, October 2012.

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

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

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

[UTF8-EXPLOIT] Davis, M. and M. Suignard, "Unicode Technical Report #36: Unicode Security Considerations", The Unicode Consortium, July 2012.

[IPFIX-MED-PROTO] Claise, B., Kobayashi, A., and B. Trammell, "Specification of the Protocol for IPFIX Mediations", draft-ietf-ipfix-mediation-protocol-03, Work in Progress, January 2013.

Acknowledgments

We would like to thank Ganesh Sadasivan, as well, for his significant contribution during the initial phases of the protocol specification. Additional thanks to Juergen Quittek for the coordination job within IPFIX and PSAMP; Nevil Brownlee, Dave Plonka, and Andrew Johnson for the thorough reviews; Randall Stewart and Peter Lei for their SCTP expertise and contributions; Martin Djernaes for the first essay on the SCTP section; Michael Behringer and Eric Vyncke for their advice and knowledge in security; Michael Tuexen for his help regarding the DTLS section; Elisa Boschi for her contribution regarding the improvement of SCTP sections; Mark Fullmer, Sebastian Zander, Jeff Meyer, Maurizio Molina, Carter Bullard, Tal Givoly, Lutz Mark, David Moore, Robert Lowe, Paul Calato, Andrew Feren, Gerhard Muenz, and many more, for the technical reviews and feedback.

Authors' Addresses

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

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

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

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

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

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

Contributors' Addresses

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

Phone: +44 (0)20 8824-8828
EMail: stbryant@cisco.com

Simon Leinen
SWITCH
Werdstrasse 2
P.O. Box
8021 Zurich
Switzerland

Phone: +41 44 268 1536
EMail: simon.leinen@switch.ch

Thomas Dietz
NEC Europe Ltd.
NEC Laboratories Europe
Network Research Division
Kurfuersten-Anlage 36
69115 Heidelberg
Germany

Phone: +49 6221 4342-128
EMail: Thomas.Dietz@nw.neclab.eu