Network Working Group
Request for Comments: 5153
Category: Informational

E. Boschi
Hitachi Europe
L. Mark
Fraunhofer FOKUS
J. Quittek
M. Stiemerling
NEC
P. Aitken
Cisco Systems, Inc.
April 2008

# IP Flow Information Export (IPFIX) Implementation Guidelines

## Status of This Memo



## Abstract

The IP Flow Information Export (IPFIX) protocol defines how IP Flow information can be exported from routers, measurement probes, or other devices. This document provides guidelines for the implementation and use of the IPFIX protocol. Several sets of guidelines address Template management, transport-specific issues, implementation of Exporting and Collecting Processes, and IPFIX implementation on middleboxes (such as firewalls, network address translators, tunnel endpoints, packet classifiers, etc.).

## Table of Contents

# 1. Introduction

The IPFIX protocol [RFC5101] defines how IP Flow information can be exported from routers, measurement probes, or other devices. In this document, we provide guidelines for its implementation.

The guidelines are split into seven main sets. These sets address implementation aspects for Template management, Exporting Process, Collecting Process, transport, implementation on middleboxes, security, and extending the information model.

Finally, this document contains a list of common mistakes related to issues that had been misinterpreted in the first IPFIX implementations and that created (and still might create) interoperability problems.

## 1.1. IPFIX Documents Overview

The IPFIX protocol [RFC5101] provides network administrators with access to IP Flow information. The architecture for the export of measured IP Flow information out of an IPFIX Exporting Process to a Collecting Process is defined in the IPFIX architecture [IPFIX-ARCH], per the requirements defined in [RFC3917].

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

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

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

## 1.2. Overview of the IPFIX Protocol

In the IPFIX protocol, { type, length, value } tuples are expressed in Templates containing { type, length } pairs, specifying which { value } fields are present in Data Records conforming to the Template, giving great flexibility as to what data is transmitted.

Since Templates are sent very infrequently compared with Data Records, this results in significant bandwidth savings.

Different Data Records may be transmitted simply by sending new Templates specifying the { type, length } pairs for the new data format. See [RFC5101] for more information.

The IPFIX information model [RFC5102] defines a large number of standard Information Elements that provide the necessary { type } information for Templates.

The use of standard elements enables interoperability among different vendors' implementations. The list of standard elements may be extended in the future through the process defined in Section 9, below. Additionally, non-standard enterprise-specific elements may be defined for private use.

## 2. Terminology

The terminology used in this document is fully aligned with the terminology defined in [RFC5101]. Therefore, the terms defined in the IPFIX terminology are capitalized in this document, as in other IPFIX documents ([RFC5101], [RFC5102], [IPFIX-ARCH]).

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

This document is Informational. It does not specify a protocol and does not use RFC 2119 key words [RFC2119] such as "MUST" and "SHOULD", except in quotations and restatements from the IPFIX standards documents. The normative specification of the protocol is given in the IPFIX protocol [RFC5101] and information model [RFC5102] documents.

## 3. Template Management Guidelines

### 3.1. Template Management

The Exporting Process should always endeavor to send Template Records before the related Data Records. However, since the Template Record may not arrive before the corresponding Data Records, the Collecting Process MAY store Data Records with an unknown Template ID pending the arrival of the corresponding Template (see Section 9 of [RFC5101]). If no Template becomes available, we recommend logging the event and discarding the corresponding Data Records, and for SCTP and TCP we recommend resetting the Transport Session. The amount of time the Collecting Process waits for a Template before resetting should be configurable. We recommend a default of 30 minutes. Note

that when using UDP as the transport protocol, this delay should be bound, when possible, by the Template Retransmit and the Template Expiry times (see Section 6.2).

The Exporting Process must be able to resend active Templates. Templates must be resent in the case of a Stream Control Transport Protocol (SCTP) association restart, a User Datagram Protocol (UDP) template refresh, or a Transmission Control Protocol (TCP) connection restart.

The Exporting Process is responsible for the management of Template IDs. Should an insufficient number of Template IDs be available, the Exporting Process must send a Template Withdrawal Message in order to free up the allocation of unused Template IDs. Note that UDP doesn't use the Template Withdrawal Message, and the Template lifetime on the Collecting Process relies on timeout.

## 3.2. Template Records versus Options Template Records

The IPFIX protocol [RFC5101] defines and specifies the use of Templates and Options Templates. Templates define the layout of Data Records, which represent Flow data. Options Templates additionally specify scope Information Elements, which can be used to define scoped Data Records. Scoped Data Records generally export control plane data (such as metadata about processes in the IPFIX collection architecture) or data otherwise applicable to multiple Flow Data Records (such as common properties as in [IPFIX-REDUCING]).

Aside from Section 4 of [RFC5101], which defines specific Options Templates to use for reporting Metering Process and Exporting Process statistics and configuration information, the choice to use Options Templates is left up to the implementer. Indeed, there is a trade-off between bandwidth efficiency and complexity in the use of Options Templates and scoped Data Records.

For example, control plane information about an Observation Point could be exported with every Flow Record measured at that Observation Point, or in a single Data Record described by an Options Template, scoped to the Observation Point identifier. In the former case, simplicity of decoding the data is gained in exchange for redundant export of the same data with every applicable Flow Record. The latter case is more bandwidth-efficient, but at the expense of requiring the Collecting Process to maintain the relationship between each applicable Flow Record and the Observation Point.

A generalized method of using Options Templates to increase bandwidth efficiency is fully described in [IPFIX-REDUCING].

### 3.3. Using Scopes

The root scope for all IPFIX Messages is the Observation Domain, which appears in the Message Header. In other words, all Data Records within a message implicitly belong to the Observation Domain. All Data Records described by Options Templates (and only those) must be restricted to an additional scope within the Observation Domain, as defined by the scope Information Elements in the Options Template Record.

In IPFIX, any Information Element can be used for scope. However, Information Elements such as counters, timestamps, padding elements, Flow properties like timeout, Flow end reason, duration, or Min/Max Flow properties [RFC5102] may not be appropriate.

Note that it is sometimes necessary to export information about entities that exist outside any Observation Domain, or within multiple Observation Domains (e.g., information about Metering Processes scoped to meteringProcessID). Such information SHOULD be exported in an IPFIX Message with Observation Domain ID 0 (see [RFC5101], Section 3.1).

### 3.4. Multiple Information Elements of the Same Type

The Exporting Process and Collecting Process MUST support the use of multiple Information Elements of the same type in a single Template [RFC5101]. This was first required by Packet Sampling (PSAMP) [PSAMP-PROTO] for the export of multiple Selector IDs. Note that the IPFIX protocol recommends that Metering Processes SHOULD use packet treatment order when exporting multiple Information Elements of the same type in the same record ([RFC5101] Section 8). This implies that ordering is important, and changes to the order of multiple identical Information Elements could cause information loss. Therefore, we strongly recommend preservation of the order of multiple Information Elements of the same type by Exporting and Collecting Processes for correct processing and storage.

### 3.5. Selecting Message Size

Section 10.3.3 of the IPFIX protocol defines the maximum message size for IPFIX Messages transported over UDP to be constrained by the path MTU, or if the path MTU is not available, 512 bytes, which is the minimum datagram size all IP implementations must support (see also Section 8.4). However, no maximum message size is imposed on other transport protocols, beyond the 65535-byte limit imposed by the 16-bit Message Length field in the IPFIX Message Header specified in Section 3.1 of [RFC5101].

An IPFIX Exporting Process operating over SCTP or TCP may export IPFIX Messages up to this 64-kB limit, and an IPFIX Collecting Process must accept any IPFIX Message up to that size.

# 4. Exporting Process Guidelines

## 4.1. Sets

A Set is identified by a Set ID [RFC5101]. A Set ID has an integral data type and its value is in the range of 0-65535. The Set ID values of 0 and 1 are not used for historical reasons [RFC3954]. A value of 2 identifies a Template Set. A value of 3 identifies an Options Template Set. Values from 4 to 255 are reserved for future use. Values above 255 are used for Data Sets. In this case, the Set ID corresponds to the Template ID of the used Template.

A Data Set received with an unknown Set ID may be stored pending the arrival of the corresponding Template (see Section 9 of [RFC5101]). If no Template becomes available, we recommend logging the event and discarding the corresponding Data Records, and for SCTP and TCP we recommend resetting the Transport Session. The amount of time the Collecting Process waits for a Template before resetting should be configurable. We recommend a default of 30 minutes. Note that when using UDP as the transport protocol, this delay should be bound, when possible, by the Template Retransmit and the Template Expiry times (see Section 6.2).

The arrival of a Set with a reserved Set ID should be logged, and the Collector must ignore the Set.

## 4.2. Information Element Coding

[IPFIX-ARCH] does not specify which entities are responsible for the encoding and decoding of Information Elements transferred via IPFIX. An IPFIX device can do the encoding either within the Metering Process or within the Exporting Process. The decoding of the Information Elements can be done by the Collecting Process or by the data processing application.

If an IPFIX node simply relays IPFIX Records (like a proxy), then no decoding or encoding of Information Elements is needed. In this case, the Exporting Process may export unknown Information Elements, i.e., Information Elements with an unknown Information Element identifier.

### 4.3. Using Counters

IPFIX offers both Delta and Total counters (e.g., octetDeltaCount, octetTotalCount). If information about a Flow is only ever exported once, then it's not important whether Delta or Total counters are used. However, if further information about additional packets in a Flow is exported after the first export, then either:

- the metering system must reset its counters to zero after the first export and report the new counter values using Delta counters, or
- the metering system must carefully maintain its counters and report the running total using Total counters.

At first, reporting the running total may seem to be the obvious choice. However, this requires that the system accurately maintains information about the Flow over a long time without any loss or error, because when reported to a Collecting Process, the previous total values will be replaced with the new information.

Delta counters offer some advantages: information about Flows doesn't have to be permanently maintained, and any loss of information has only a small impact on the total stored at the Collecting Process. Finally, Deltas may be exported in fewer bytes than Total counters using the IPFIX "Reduced Size Encoding" scheme [RFC5101].

Note that Delta counters have an origin of zero and that a Collecting Process receiving Delta counters for a Flow that is new to the Collecting Process must assume the Deltas are from zero.

### 4.4. Padding

The IPFIX information model defines an Information Element for padding called paddingOctets [RFC5102]. It is of type octetArray, and the IPFIX protocol allows encoding it as a fixed-length array as well as a variable-length array.

The padding Information Element can be used to align Information Elements within Data Records, Records within Sets, and Sets within IPFIX Messages, as described below.

#### 4.4.1. Alignment of Information Elements within a Data Record

The padding Information Element gives flexible means for aligning Information Elements within a Data Record. Aligning within a Data Record can be useful, because internal data structures can be easily converted into Flow Records at the Exporter and vice versa at the Collecting Process.

Alignment of Information Elements within a Data Record is achieved by inserting an instance of the paddingOctets Information Element with appropriate length before each unaligned Information Element. This insertion is explicitly specified within the Template Record or Options Template Record, respectively, that corresponds to the Data Record.

#### 4.4.2. Alignment of Information Element Specifiers within a Template Record

There is no means for aligning Information Element specifiers within Template Records. However, there is limited need for such a method, as Information Element specifiers are always 32-bit aligned, and 32-bit alignment is generally sufficient.

#### 4.4.3. Alignment of Records within a Set

There is no means for aligning Template Records within a Set. However, there is limited need for such a method, as Information Element specifiers are always 32-bit aligned, and 32-bit alignment is generally sufficient.

Data Records can be aligned within a Set by appending instances of the paddingOctets Information Element at the end of the Record. Since all Data Records within a Set have the same structure and size, aligning one Data Record implies aligning all the Data Records within a single Set.

#### 4.4.4. Alignment of Sets within an IPFIX Message

If Records are already aligned within a Set by using paddingOctets Information Elements, then this alignment will already be achieved. But for aligning Sets within an IPFIX Message, padding Information Elements can be used at the end of the Set so that the subsequent Set starts at an aligned boundary. This padding mechanism is described in Section 3.3.1 of [RFC5101] and can be applied even if the Records within the Set are not aligned. However, it should be noted that this method is limited by the constraint that "the padding length MUST be shorter than any allowable Record in the Set", to prevent the padding from being misinterpreted as an additional Data Record.

### 4.5. Time Issues

IPFIX Messages contain the export time in the Message Header. In addition, there is a series of Information Elements defined to transfer time values. [RFC5102] defines four abstract data types to transfer time values in second, millisecond, microsecond, and nanosecond resolution.

The accuracy and precision of these values depend on the accuracy and the precision of the Metering Process clock. The accuracy and precision of the Exporting Process clock, and the synchronization of the Metering Process and Exporting Process clocks, are also important when using the delta timestamp Information Elements. To ensure accuracy, the clocks should be synchronized to a UTC time source. Normally, it would be sufficient to derive the time from a remote time server using the Network Time Protocol (NTP) [RFC1305]. IPFIX Devices operating with time values of microsecond or nanosecond resolution need direct access to a time source, for example, to a GPS (Global Positioning System) unit.

The most important consideration in selecting timestamp Information Elements is to use a precision appropriate for the timestamps as generated from the Metering Process. Specifically, an IPFIX Device should not export timestamp Information Elements of higher precision than the timestamps used by the Metering Process (e.g., millisecond-precision Flows should not be exported with flowStartMicroseconds and flowEndMicroseconds).

### 4.6. IPFIX Message Header Export Time and Data Record Time

Section 5 of [RFC5101] defines a method for optimized export of time-related Information Elements based upon the Export Time field of the IPFIX Message Header. The architectural separation of the Metering Process and Exporting Process in [IPFIX-ARCH] raises some difficulties with this method, of which implementers should be aware.

Since the Metering Process has no information about the export time of the IPFIX Message (that is, when the message leaves the Exporting Process), it cannot properly use the delta time Information Elements; it must store absolute timestamps and transmit these to the Exporting Process. The Exporting Process must then convert these to delta timestamps once the export time is known. This increases the processing burden on the Exporting Process. Note also that the absolute timestamps require more storage than their delta timestamp counterparts. However, this method can result in reduced export bandwidth.

Alternatively, the Exporting Process may simply export absolute timestamp Information Elements. This simplifies the Exporting Process' job and reduces processing burden, but increases export bandwidth requirements.

### 4.7. Devices without an Absolute Clock

Exporting just relative times in a device without an absolute clock is often not sufficient. For instance, observed traffic could be retained in the device's cache for some time before being exported (e.g., if the Exporter runs once per minute), or stuck in an Inter Process Communication (IPC) queue, or stuck in the export stack, or delayed in the network between the Exporter and Collector.

For these reasons, it can be difficult for the Collecting Process to convert the relative times exported using the flowStartSysUpTime and flowEndSysUpTime Information Elements to absolute times with any sort of accuracy without knowing the systemInitTimeMilliseconds. Therefore, the sending of the flowStartSysUpTime and flowEndSysUpTime Information Elements without also sending the systemInitTimeMilliseconds Information Element is not recommended.

## 5. Collecting Process Guidelines

### 5.1. Information Element (De)Coding

Section 9 of [RFC5101] specifies: "The Collecting Process MUST note the Information Element identifier of any Information Element that it does not understand and MAY discard that Information Element from the Flow Record". The Collecting Process may accept Templates with Information Elements of unknown types. In this case, the value received for these Information Elements should be decoded as an octet array.

Alternatively, the Collecting Process may ignore Templates and subsequent Data Sets that contain Information Elements of unknown types.

It is recommended that Collecting Processes provide means to flexibly add types of new Information Elements to their knowledge base. An example is a configuration file that is read by the Collecting Process and that contains a list of Information Element identifiers and their corresponding types. Particularly for adding enterprise-specific Information Elements, such a feature can be very useful.

### 5.2. Reduced-Size Encoding of Information Elements

Since a Collector may receive data from the same device and Observation Domain in two Templates using different reduced-size encodings, it is recommended that the data be stored using full-size encoding, to ensure that the values can be stored or even aggregated together.

### 5.3. Template Management

Template IDs are generated dynamically by the Exporting Process. They are unique per Transport Session and Observation Domain.

Therefore, for each Transport Session, the Collecting Process has to maintain a list of Observation Domains. For each Observation Domain, the Collecting Process has to maintain a list of current Template IDs in order to decode subsequent Data Records.

Note that a restart of the Transport Session may lead to a Template ID renumbering.

## 6. Transport-Specific Guidelines

IPFIX can use SCTP, TCP, or UDP as a transport protocol. IPFIX implementations MUST support SCTP with partial reliability extensions (PR-SCTP), and MAY support TCP and/or UDP (see [RFC5101], Section 10.1). In the IPFIX documents, the terms SCTP and PR-SCTP are often used interchangeably to mean SCTP with partial reliability extensions.

### 6.1. SCTP

PR-SCTP is the preferred transport protocol for IPFIX because it is congestion-aware, reducing total bandwidth usage in the case of congestion, but with a simpler state machine than TCP. This saves resources on lightweight probes and router line cards.

SCTP, as specified in [RFC4960] with the PR-SCTP extension defined in [RFC3758], provides several features not available in TCP or UDP. The two of these most universally applicable to IPFIX implementations, and which IPFIX implementers need to know about, are multiple streams and per-message partial reliability.

An SCTP association may contain multiple streams. Streams are useful for avoiding head-of-line blocking, thereby minimizing end-to-end delay from the Exporting Process to the Collecting Process. Example

applications for this feature would be using one SCTP stream per Observation Domain, one stream per type of data (or Template ID), or one stream for Flow data and one for metadata.

An Exporting Process may request any number of streams, and may send IPFIX Messages containing any type of Set (Data Set, Template Set, etc.) on any stream. A Collecting Process MUST be able to process any Message received on any stream.

Stream negotiation is a feature of the SCTP protocol. Note, however, that the IPFIX protocol doesn't provide any mechanism for the Exporter to convey any information about which streams are in use to the Collector. Therefore, stream configuration must be done out of band.

One extra advantage of the PR-SCTP association is its ability to send messages with different levels of reliability, selected according to the application. For example, billing or security applications might require reliable delivery of all their IPFIX Messages, while capacity planning applications might be more tolerant of message loss. SCTP allows IPFIX Messages for all these applications to be transported over the same association with the appropriate level of reliability.

IPFIX Messages may be sent with full or partial reliability, on a per-message basis. Fully reliable delivery guarantees that the IPFIX Message will be received at the Collecting Process or that that SCTP association will be reset, as with TCP. Partially reliable delivery does not guarantee the receipt of the IPFIX Message at the Collecting Process. This feature may be used to allow Messages to be dropped during network congestion, i.e., while observing a Denial of Service attack.

[RFC3758] defines the concept of a Partial Reliability policy, which specifies the interface used to control partially reliable delivery. It also defines a single example Partial Reliability policy called "timed reliability", which uses a single parameter: lifetime. The lifetime is specified per message in milliseconds, and after it expires, no further attempt will be made to transmit the message. Longer lifetimes specify more retransmission attempts per message and therefore higher reliability; however, it should be noted that the absolute reliability provided by a given lifetime is highly dependent on network conditions, so an Exporting Process using the timed reliability service should provide a mechanism for configuring the lifetime of exported IPFIX Messages. Another possible Partial Reliability policy could be limited retransmission, which guarantees a specified number of retransmissions for each message. It is up to the implementer to decide which Partial Reliability policy is most appropriate for its application.

There is an additional service provided by SCTP and useful in conjunction with PR-SCTP: unordered delivery. This also works on a per-message basis by declaring that a given message should be delivered to the receiver as soon as it is queued rather than kept in sequence; however, it should be noted that unless explicitly requested by the sender, even messages sent partially reliably will still be delivered in order. Unordered delivery should not be used when the order of IPFIX Messages may matter: e.g., a Template or Options Template. Unordered delivery should not be used when Total counters are used, as reordering could result in the counter value decreasing at the Collecting Process and even being left with a stale value if the last message processed is stale.

By convention, when the IPFIX documents state a requirement for reliable delivery (as, for example, the IPFIX protocol document does for Template Sets, Options Template Sets, and Template Withdrawal Messages), an IPFIX Exporting Process must not use partially reliable delivery for those Messages. By default, and explicitly if the IPFIX documents call for "partially reliable" or "unreliable" delivery, an IPFIX Exporting Process may use partially reliable delivery if the other requirements of the application allow.

The Collecting Process may check whether IPFIX Messages are lost by checking the Sequence Number in the IPFIX header. The Collecting Process should use the Sequence Number in the IPFIX Message Header to determine whether any messages are lost when sent with partial reliability. Sequence Numbers should be tracked independently for each stream.

The following may be done to mitigate message loss:

- Increase the SCTP buffer size on the Exporter.

- Increase the bandwidth available for communicating the exported Data Records.

- Use sampling, filtering, or aggregation in the Metering Process to reduce the amount of exported data (see [RFC5101], Section 10.4.2.3).

- If partial reliability is used, switch to fully reliable delivery on the Exporting Process or increase the level of partial reliability (e.g., when using timed reliability, by specifying a longer lifetime for exported IPFIX Messages).

If the SCTP association is brought down because the IFPIX Messages can't be exported reliably, the options are:

- Increase the SCTP buffer size on the Exporter.
- Increase the bandwidth available for communicating the exported Data Records.
- Use sampling, filtering, or aggregation in the Metering Process to reduce the amount of exported data.

Note that Templates must not be resent when using SCTP, without an intervening Template Withdrawal or SCTP association reset. Note also that since Template Sets and Template Withdrawal Messages may be sent on any SCTP stream, a Template Withdrawal Message may withdraw a Template sent on a different stream, and a Template Set may reuse a Template ID withdrawn by a Template Withdrawal Message sent on a different stream. Therefore, an Exporting Process sending Template Withdrawal Messages should ensure to the extent possible that the Template Withdrawal Messages and subsequent Template Sets reusing the withdrawn Template IDs are received and processed at the Collecting Process in proper order. The Exporting Process can achieve this by one of two possible methods: 1. by sending a Template Withdrawal Message reliably, in order, and on the same stream as the subsequent Template Set reusing its ID; or 2. by waiting an appropriate amount of time (on the scale of one minute) after sending a Template Withdrawal Message before attempting to reuse the withdrawn Template ID.

### 6.2. UDP

UDP is useful in simple systems where an SCTP stack is not available, and where there is insufficient memory for TCP buffering.

However, UDP is not a reliable transport protocol, and IPFIX Messages sent over UDP might be lost as with partially reliable SCTP streams. UDP is not the recommended protocol for IPFIX and is intended for use in cases in which IPFIX is replacing an existing NetFlow infrastructure, with the following properties:

- A dedicated network,
- within a single administrative domain,
- where SCTP is not available due to implementation constraints, and
- the Collector is as topologically close as possible to the Exporter.

Note that because UDP itself provides no congestion control mechanisms, it is recommended that UDP transport be used only on managed networks, where the network path has been explicitly provisioned for IPFIX traffic through traffic engineering mechanisms, such as rate limiting or capacity reservations.

An important example of an explicitly provisioned, managed network for IPFIX is the use of IPFIX to replace a functioning NetFlow implementation on a dedicated network. In this situation, the dedicated network should be provisioned in accordance with the NetFlow deployment experience that Flow export traffic generated by monitoring an interface will amount to 2-5% of the monitored interface's bandwidth.

As recommended in [TSVWG-UDP], an application should not send UDP messages that result in IP packets that exceed the MTU of the path to the destination and should enable UDP checksums (see Sections 3.2 and 3.4 of [TSVWG-UDP], respectively).

Since IPFIX assumes reliable transport of Templates over SCTP, this necessitates some changes for IPFIX Template management over UDP. Templates sent from the Exporting Process to the Collecting Process over UDP MUST be resent at regular time intervals; these intervals MUST be configurable (see Section 10.3 of [RFC5101]).

We recommend a default Template-resend time of 10 minutes, configurable between 1 minute and 1 day.

Note that this could become an interoperability problem; e.g., if an Exporter resends Templates once per day, while a Collector expires Templates hourly, then they may both be IPFIX-compatible, but not be interoperable.

Retransmission time intervals that are too short waste bandwidth on unnecessary Template retransmissions. On the other hand, time intervals that are too long introduce additional costs or risk of data loss by potentially requiring the Collector to cache more data without having the Templates available to decode it.

To increase reliability and limit the amount of potentially lost data, the Exporting Process may resend additional Templates using a packet-based schedule. In this case, Templates are resent depending on the number of data packets sent. Similarly to the time interval, resending a Template every few packets introduces additional overhead, while resending after a large amount of packets have already been sent means high costs due to the data caching and potential data loss.

We recommend a default Template-resend interval of 20 packets, configurable between 1 and 1000 data packets.

Note that a sufficiently small resend time or packet interval may cause a system to become stuck, continually resending Templates or Options Data. For example, if the resend packet interval is 2 (i.e., Templates or Options Data are to be sent in every other packet) but more than two packets are required to send all the information, then the resend interval will have expired by the time the information has been sent, and Templates or Options Data will be sent continuously -- possibly preventing any data from being sent at all. Therefore, the resend intervals should be considered from the last data packet, and should not be tied to specific Sequence Numbers.

The Collecting Process should use the Sequence Number in the IPFIX Message Header to determine whether any messages are lost.

The following may be done to mitigate message loss:

- Move the Collector topologically closer to the Exporter.
- Increase the bandwidth of the links through which the Data Records are exported.
- Use sampling, filtering, or aggregation in the Metering Process to reduce the amount of exported data.
- Increase the buffer size at the Collector and/or the Exporter.

Before using a Template for the first time, the Exporter may send it in several different IPFIX Messages spaced out over a period of packets in order to increase the likelihood that the Collector has received the Template.

Template Withdrawal Messages MUST NOT be sent over UDP (per Section 10.3.6 of [RFC5101]). The Exporter must rely on expiration at the Collector to expire old Templates or to reuse Template IDs.

We recommend that the Collector implements a Template Expiry of three times the Exporter refresh rate.

However, since the IPFIX protocol doesn’t provide any mechanism for the Exporter to convey any information about the Template Expiry time to the Collector, configuration must be done out of band.

If no out-of-band configuration is made, we recommend to initially set a Template Expiry time at the Collector of 60 minutes. The Collecting Process may estimate each Exporting Process's resend time and adapt the Expiry time for the corresponding Templates accordingly.

### 6.3. TCP

TCP can be used as a transport protocol for IPFIX if one of the endpoints has no support for SCTP, but a reliable transport is needed and/or the network between the Exporter and the Collector has not explicitly been provisioned for the IPFIX traffic. TCP is one of the core protocols of the Internet and is widely supported.

The Exporting Process may resend Templates (per UDP, above), but it's not required to do so, per Section 10.4.2.2 of [RFC5101]:

"A Collecting Process MUST record all Template and Options Template Records for the duration of the connection, as an Exporting Process is not required to re-export Template Records."

If the available bandwidth between Exporter and Collector is not sufficient or the Metering Process generates more Data Records than the Collector is capable of processing, then TCP congestion control may cause the Exporter to block. Options in this case are:

- Increase the TCP buffer size on the Exporter.
- Increase the bandwidth of the links through which the Data Records are exported.
- Use sampling, filtering, or aggregation in the Metering Process to reduce the amount of exported data.

## 7. Guidelines for Implementation on Middleboxes

The term middlebox is defined in [RFC3234] as:

"any intermediary device performing functions other than the normal, standard functions of an IP router on the datagram path between a source host and destination host."

The list of middleboxes discussed in [RFC3234] contains:

1. Network Address Translation (NAT),
2. NAT-Protocol Translation (NAT-PT),

3. SOCKS gateway,
4. IP tunnel endpoints,
5. packet classifiers, markers, schedulers,
6. transport relay,
7. TCP performance enhancing proxies,
8. load balancers that divert/munge packets,
9. IP firewalls,
10. application firewalls,
11. application-level gateways,
12. gatekeepers / session control boxes,
13. transcoders,
14. proxies,
15. caches,
16. modified DNS servers,
17. content and applications distribution boxes,
18. load balancers that divert/munge URLs,
19. application-level interceptors,
20. application-level multicast,
21. involuntary packet redirection,
22. anonymizers.

It is likely that since the publication of RFC 3234 new kinds of middleboxes have been added.

While the IPFIX specifications [RFC5101] based the requirements on the export protocol only (as the IPFIX name implies), these sections cover the guidelines for the implementation of the Metering Process by recommending which Information Elements to export for the different middlebox considerations.

### 7.1. Traffic Flow Scenarios at Middleboxes

Middleboxes may delay, reorder, drop, or multiply packets; they may change packet header fields and change the payload. All these actions have an impact on traffic Flow properties. In general, a middlebox transforms a unidirectional original traffic Flow T that arrives at the middlebox into a transformed traffic Flow T' that leaves the middlebox.


Figure 1: Unidirectional traffic Flow traversing a middlebox

Note that in an extreme case, T' may be an empty traffic Flow (a Flow with no packets), for example, if the middlebox is a firewall and blocks the Flow.

In case of a middlebox performing a multicast function, a single original traffic Flow may be transformed into more than one transformed traffic Flow.

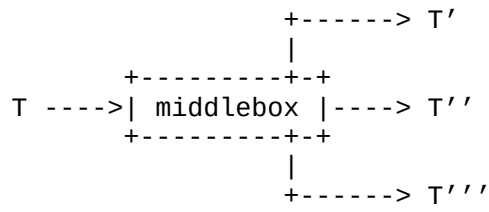


Figure 2: Unidirectional traffic Flow traversing a middlebox with multicast function

For bidirectional traffic Flows, we identify Flows on different sides of the middlebox; say, T_l on the left side and T_r on the right side.

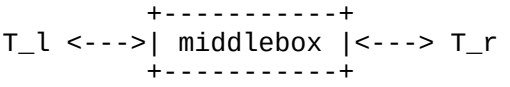


Figure 3: Bidirectional unicast traffic Flow traversing a middlebox

In case of a NAT, T_l might be a traffic Flow in a private address realm and T_r the translated traffic Flow in the public address realm. If the middlebox is a NAT-PT, then T_l may be an IPv4 traffic Flow and T_r the translated IPv6 traffic Flow.

At tunnel endpoints, Flows are multiplexed or demultiplexed. In general, tunnel endpoints can deal with bidirectional traffic Flows.

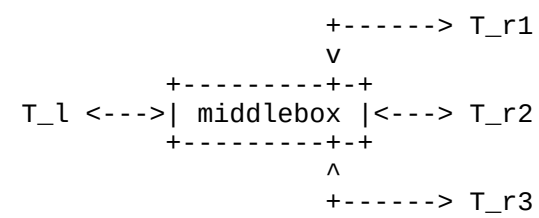


Figure 4: Multiple data reduction

An example is a traffic Flow T_l of a tunnel and Flows T_rx that are multiplexed into or demultiplexed out of a tunnel. According to the IPFIX definition of traffic Flows in [RFC5101], T and T' or T_l and T_rx, respectively, are different Flows in general.

However, from an application point of view, they might be considered as closely related or even as the same Flow, for example, if the payloads they carry are identical.

## 7.2. Location of the Observation Point

Middleboxes might be integrated with other devices. An example is a router with a NAT or a firewall at a line card. If an IPFIX Observation Point is located at the line card, then the properties of measured traffic Flows may depend on the side of the integrated middlebox at which packets were captured for traffic Flow measurement.

Consequently, an Exporting Process reporting traffic Flows measured at a device that hosts one or more middleboxes should clearly indicate to Collecting Processes the location of the used Observation Point(s) with respect to the middlebox(es). This can be done by using Options with Observation Point as scope and elements like, for instance, lineCardID or samplerID. Otherwise, processing the measured Flow data could lead to wrong results.

At first glance, choosing an Observation Point that covers the entire middlebox looks like an attractive choice. But this leads to ambiguities for all kinds of middleboxes. Within the middlebox, properties of packets are modified, and it should be clear at a Collecting Process whether packets were observed and metered before or after modification. For example, it must be clear whether a reported source IP address was observed before or after a NAT changed it or whether a reported packet count was measured before or after a

firewall dropped packets. For this reason, [RFC5102] provides Information Elements with prefix "post" for Flow properties that are changed within a middlebox.

If an Observation Point is located inside a middlebox, the middlebox must have well-defined and well-separated internal functions, for example, a combined NAT and firewall, and the Observation Point should be located on a boundary between middlebox functions rather than within one of the functions.

### 7.3. Reporting Flow-Related Middlebox Internals

While this document recommends IPFIX implementations using Observation Points outside of middlebox functions, there are a few special cases where reporting Flow-related internals of a middlebox is of interest.

For many applications that use traffic measurement results, it is desirable to get more information than can be derived from just observing packets on one side of a middlebox. If, for example, packets are dropped by the middlebox acting as a firewall, NAT, or traffic shaper, then information about how many observed packets are dropped may be of high interest.

This section gives recommendations on middlebox internal information that may be reported if the IPFIX Observation Point is co-located with one or more middleboxes. Since the internal information to be reported depends on the kind of middlebox, it is discussed per kind.

The recommendations cover middleboxes that act per packet and that do not modify the application-level payload of the packet (except by dropping the entire packet) and that do not insert additional packets into an application-level or transport-level traffic stream.

Covered are the packet-level middleboxes of kinds 1, 2, 3, 5, 9, 10, 21, and 22 (according to the enumeration given at the beginning of Section 7 of this document). Not covered are 4, 6-8 and 11-20. TCP performance-enhancing proxies (7) are not covered because they may add ACK packets to a TCP connection.

Still, if possible, IPFIX implementations co-located with uncovered middleboxes (i.e., of type 7 or 11-20) should follow the recommendations given in this section if they can be applied in a way that reflects the intention of these recommendations.

#### 7.3.1. Packet Dropping Middleboxes

If an IPFIX Observation Point is co-located with one or more middleboxes that potentially drop packets, then the corresponding IPFIX Exporting Process should be able to report the number of packets that were dropped per reported Flow.

Concerned kinds of middleboxes are NAT (1), NAT-PT (2), SOCKS gateway (3), packet schedulers (5), IP firewalls (9) and application-level firewalls (10).

#### 7.3.2. Middleboxes Changing the DSCP

If an IPFIX Observation Point is co-located with one or more middleboxes that potentially modify the Diffserv Code Point (DSCP, see [RFC2474]) in the IP header, then the corresponding IPFIX Exporting Process should be able to report both the observed incoming DSCP value and also the DSCP value on the 'other' side of the middlebox (if this is a constant value for the particular traffic flow). The related Information Elements specified in [RFC5102] are: IpClassOfService and postIpClassOfService.

Note that the current IPFIX information model only contains Information Elements supporting packets observed before the DSCP change, i.e. ipClassOfService and postIpClassOfService, where the latter reports the value of the IP TOS field after the DSCP change. We recommend, whenever possible, to move the Observation Point to the point before the DSCP change and report the Observed and post- values. If reporting the value of the IP TOS field before DSCP change is required, "pre" values can be exported using enterprise-specific Information Elements.

Note also that a classifier may change the same DSCP value of packets from the same Flow to different values depending on the packet or other conditions. Also, it is possible that packets of a single unidirectional arriving Flow contain packets with different DSCP values that are all set to the same value by the middlebox. In both cases, there is a constant value for the DSCP field in the IP packet header to be observed on one side of the middlebox, but on the other side the value may vary. In such a case, reliable reporting of the DSCP value on the 'other' side of the middlebox is not possible by just reporting a single value. According to the IPFIX information model [RFC5102], the first value observed for the DSCP is reported by the IPFIX protocol in that case.

This recommendation applies to packet markers (5).

### 7.3.3. Middleboxes Changing IP Addresses and Port Numbers

If an IPFIX Observation Point is co-located with one or more middleboxes that potentially modify the:

- IP version field,
- IP source address header field,
- IP destination address header field,
- Source transport port number, or
- Destination transport port number

in one of the headers, then the corresponding IPFIX Exporting Process should be able to report the 'translated' value of these fields, as far as they have constant values for the particular traffic Flow, in addition to the observed values of these fields.

If the changed values are not constant for the particular traffic Flow but still reporting is desired, then it is recommended that the general rule from [RFC5102] for Information Elements with changing values is applied: the reported value is the one that applies to the first packet observed for the reported Flow.

Note that the 'translated' value of the fields can be the values before or after the translation depending on the Flow direction and the location of the Observation Point with respect to the middlebox. We always call the value that is not the one observed at the Observation Point the translated value.

Note also that a middlebox may change the same port number value of packets from the same Flow to different values depending on the packet or other conditions. Also, it is possible that packets of different unidirectional arriving Flows with different source/destination port number pairs may be mapped to a single Flow with a single source/destination port number pair by the middlebox. In both cases, there is a constant value for the port number pair to be observed on one side of the middlebox, but on the other side the values may vary. In such a case, reliable reporting of the port number pairs on the 'other' side of the middlebox is not possible. According to the IPFIX information model [RFC5102], the first value observed for each port number is reported by the IPFIX protocol in that case.

This recommendation applies to NAT (1), NAT-PT (2), SOCKS gateway (3) and involuntary packet redirection (21) middleboxes. It may also be applied to anonymizers (22), though it should be noted that this carries the risk of losing the effect of anonymization.

# 8. Security Guidelines

## 8.1. Introduction to TLS and DTLS for IPFIX Implementers

Transport Layer Security (TLS) [RFC4346] and Datagram Transport Layer Security (DTLS) [RFC4347] are the REQUIRED protocols for securing network traffic exported with IPFIX (see Section 11 of [RFC5101]). TLS requires a reliable transport channel and is selected as the security mechanism for TCP. DTLS is a version of TLS capable of securing datagram traffic and is selected for UDP, SCTP, and PR-SCTP.

When mapping TLS terminology used in [RFC4346] to IPFIX terminology, keep in mind that the IPFIX Exporting Process, as it is the connection initiator, corresponds to the TLS client, and the IPFIX Collecting Process corresponds to the TLS server. These terms apply only to the bidirectional TLS handshakes done at Transport Session establishment and completion time; aside from TLS connection set up between the Exporting Process and the Collecting Process, and teardown at the end of the session, the unidirectional Flow of messages from Exporting Process to Collecting Process operates over TLS just as over any other transport layer for IPFIX.

## 8.2. X.509-Based Identity Verification for IPFIX over TLS or DTLS

When using TLS or DTLS to secure an IPFIX Transport Session, the Collecting Process and Exporting Process must use strong mutual authentication. In other words, each IPFIX endpoint must have its own X.509 certificate [RFC3280] and private key, and the Collecting Process, which acts as the TLS or DTLS server, must send a Certificate Request to the Exporting Process during the TLS handshake, and fail to establish a session if the Exporting Process does not present a valid certificate.

Each Exporting Process and Collecting Process must verify the identity of its peer against a set of authorized peers. This may be done by configuring a set of authorized distinguished names and comparing the peer certificate’s subject distinguished name against each name in the set. However, if a private certification authority (CA) is used to sign the certificates identifying the Collecting Processes and Exporting Processes, and the set of certificates signed by that private CA may be restricted to those identifying peers authorized to communicate with each other, it is sufficient to merely verify that the peer’s certificate is issued by this private CA.

When verifying the identity of its peer, an IPFIX Exporting Process or Collecting Process must verify that the peer certificate's subject common name or subjectAltName extension dNSName matches the fully-qualified domain name (FQDN) of the peer. This involves retrieving the expected domain name from the peer certificate and the address of the peer, then verifying that the two match via a DNS lookup. Such verification should require both that forward lookups (FQDN to peer address) and reverse lookups (peer address to FQDN) match. In deployments without DNS infrastructure, it is acceptable to represent the FQDN as an IPv4 dotted-quad or a textual IPv6 address as in [RFC1924].

### 8.3. Implementing IPFIX over TLS over TCP

Of the security solutions specified for IPFIX, TLS over TCP is as of this writing the most mature and widely implemented. Until stable implementations of DTLS over SCTP are widely available (see Section 8.5, below), it is recommended that applications requiring secure transport for IPFIX Messages use TLS over TCP.

When using TLS over TCP, IPFIX Exporting Processes and Collecting Processes should behave in all other aspects as if using TCP as the transport protocol, especially as regards the handling of Templates and Template withdrawals.

### 8.4. Implementing IPFIX over DTLS over UDP

An implementation of the DTLS protocol version 1, described in [RFC4347] and required to secure IPFIX over UDP, is available in OpenSSL [OPENSSL] as of version 0.9.8. However, DTLS support is as of this writing under active development and certain implementations might be unstable. We recommend extensive testing of DTLS-based IPFIX implementations to build confidence in the DTLS stack over which your implementation runs.

When using DTLS over UDP, IPFIX Exporting Processes and Collecting Processes should behave in all other aspects as if using UDP as the transport protocol, especially as regards the handling of Templates and Template timeouts.

Note that the selection of IPFIX Message sizes for DTLS over UDP must account for overhead per packet introduced by the DTLS layer.

### 8.5. Implementing IPFIX over DTLS over SCTP

As of this writing, there is no publicly available implementation of DTLS over SCTP as described in [RFC4347] and [TUEXEN].

When using DTLS over SCTP, IPFIX Exporting Processes and Collecting Processes should behave in all other aspects as if using SCTP as the transport protocol, especially as regards the handling of Templates and the use of reliable transport for Template and scope information.

An implementation of the DTLS protocol version 1, described in [RFC4347] and required to secure IPFIX over SCTP, is available in OpenSSL [OPENSSL] as of version 0.9.8. However, DTLS support is as of this writing under active development and certain implementations might be unstable. We recommend extensive testing of DTLS-based IPFIX implementations to build confidence in the DTLS stack over which your implementation runs.

## 9. Extending the Information Model

IPFIX supports two sets of Information Elements: IANA-registered Information Elements and enterprise-specific Information Elements. New Information Elements can be added to both sets as described in this section. If an Information Element is considered of general interest, it should be added to the set of IETF-specified Information Elements that is maintained by IANA.

Alternatively, private enterprises can define proprietary Information Elements for internal purposes. There are several potential reasons for doing so. For example, the Information Element might only relate to proprietary features of a device or protocol of the enterprise. Also, pre-standard product delivery or commercially sensitive product features might cause the need for enterprise-specific Information Elements.

The IPFIX information model [RFC5102] document contains an XML-based specification of Template, abstract data types, and IPFIX Information Elements, which may be used to create consistent machine-readable extensions to the IPFIX information model. This description can be used for automatically checking syntactic correctness of the specification of IPFIX Information Elements and for generating code that deals with processing IPFIX Information Elements.

### 9.1. Adding New IETF-Specified Information Elements

New IPFIX Information Elements that are considered to be of general interest should be added to the set of IETF-specified Information Elements that is maintained by IANA.

The introduction of new Information Elements in the IANA registry is subject to expert review. As described in Section 7.1 of [RFC5102], an expert review is performed by one of a group of experts designated by an IETF Operations and Management Area Director. The experts will initially be drawn from the Working Group Chairs and document editors of the IPFIX and PSAMP Working Groups. The group of experts must double check the Information Elements definitions with already defined Information Elements for completeness, accuracy, redundancy, and correct naming following the naming conventions in [RFC5102], Section 2.3.

The specification of new IPFIX Information Elements must use the Template specified in [RFC5102], Section 2.1, and must be published using a well-established and persistent publication medium.

### 9.2. Adding Enterprise-Specific Information Elements

Enterprises or other organizations holding a registered Structure of Management Information (SMI) network management private enterprise code number can specify enterprise-specific Information Elements. Their identifiers can be chosen arbitrarily within the range of 1-32767 and have to be coupled with a Private Enterprise Identifier [PEN]. Enterprise identifiers MUST be registered as SMI network management private enterprise code numbers with IANA. The registry can be found at http://www.iana.org/assignments/enterprise-numbers.

## 10. Common Implementation Mistakes

The issues listed in this section were identified during implementation and interoperability testing. They do not stem from insufficient clarity in the protocol, but each of these was an actual mistake made in a tested IPFIX implementation. They are listed here for the convenience of future implementers.

### 10.1. IPFIX and NetFlow Version 9

A large group of mistakes stems from the fact that many implementers started implementing IPFIX from an existing version of NetFlow version 9 [RFC3954]. Despite their similarity, the two protocols differ in many aspects. We list here some of the most important differences.

- Transport protocol: NetFlow version 9 initially ran over UDP, while IPFIX must have a congestion-aware transport protocol. IPFIX specifies PR-SCTP as its mandatory protocol, while TCP and UDP are optional.

- IPFIX differentiates between IANA-registered and enterprise-specific Information Elements. Enterprise-specific Information Elements can be specified by coupling a non-IANA-registered Information Element identifier with an Enterprise ID (corresponding to the vendor that defined the Information Element).

- Options Templates: in IPFIX, an Options Template must have a scope, and the scope is not allowed to be of length zero. The NetFlow version 9 specifications [RFC3954] don't specify that the scope must not be of length zero.

Message Header:

- Set ID: Even if the packet headers are different between IPFIX and NetFlow version 9, similar fields are used in both of them. The difference between the two protocols is in the values that these fields can assume. A typical example is the Set ID values: the Set ID values of 0 and 1 are used in NetFlow version 9, while they are not used in IPFIX.

- Length field: in NetFlow version 9, this field (called count) contains the number of Records. In IPFIX, it indicates the total length of the IPFIX Message, measured in octets (including Message Header and Set(s)).

- Timestamp: the NetFlow version 9 header has an additional timestamp: sysUpTime. It indicates the time in milliseconds since the last reboot of the Exporting Process.

- The version number is different. NetFlow version 9 uses the version number 9, while IPFIX uses the version number 10.

## 10.2. Padding of the Data Set

[RFC5101] specifies that the Exporting Process MAY insert some octets for set padding to align Data Sets within a Message. The padding length must be shorter than any allowable Record in that set.

It is important to respect this limitation: if the padding length is equal to or longer than the length of the shortest Record, it will be interpreted as another Record.

An alternative is to use the paddingOctets Information Element in the Template definition.

### 10.3. Field ID Numbers

Information Element numbers in IPFIX have the range 0-32767 (0-0x7FFF). Information Element numbers outside this range (i.e., with the high bit set) are taken to be enterprise-specific Information Elements, which have an additional four-byte Private Enterprise Number following the Information Element number and length. Inadvertently setting the high bit of the Information Element number by selecting a number out of this range will therefore cause Template scanning errors.

### 10.4. Template ID Numbers

Template IDs are generated as required by the Exporting Process. When the same set of Information Elements is exported at different times, the corresponding Template is usually identified by different Template IDs. Similarly, if multiple co-existing Templates are composed of the same set of Information Elements, they are also identified by different Template IDs. The Collecting Process does not know in advance which Template ID a particular Template will use.

## 11. Security Considerations

This document describes the implementation guidelines of IPFIX. The security requirements for the IPFIX target applications are addressed in the IPFIX requirements document [RFC3917]. These requirements are considered for the specification of the IPFIX protocol [RFC5101], for which a Security Considerations Section exists.

Section 7 of this document recommends that IPFIX Exporting Processes report internals about middleboxes. These internals may be security-relevant, and the reported information needs to be protected appropriately for reasons given below.

Reporting of packets dropped by firewalls and other packet-dropping middleboxes carries the risk that this information can be used by attackers for analyzing the configuration of the middlebox and for developing attacks against it. Address translation may be used for hiding the network structure behind an address translator. If an IPFIX Exporting Process reports the translations performed by an address translator, then parts of the network structure may be revealed. If an IPFIX Exporting Process reports the translations performed by an anonymizer, the main function of the anonymizer may be compromised.

Note that there exist vulnerabilities in DTLS over SCTP as specified in the IPFIX protocol, such that a third party could cause messages to be undetectably lost, or an SCTP association to shut down. These

vulnerabilities are addressed by [TUEXEN]; however, it is unclear whether initial OpenSSL-based implementations of DTLS over SCTP will contain the required fixes. DTLS over SCTP should be used with caution in production environments until these issues are completely addressed.

## 12. Acknowledgments

We would like to thank the MoMe project for organizing two IPFIX Interoperability Events in July 2005 and in March 2006, and Fraunhofer Fokus for organizing the third one in November 2006. The Interoperability Events provided us precious input for this document. Thanks to Brian Trammell for his contributions to the SCTP section and the security guidelines and for the multiple thorough reviews. We would also like to thank Benoit Claise, Carsten Schmoll, and Gerhard Muenz for the technical review and feedback, and Michael Tuexen, Randall Stewart, and Peter Lei for reviewing the SCTP section.

## 13. References

### 13.1. Normative References

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

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

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

### 13.2. Informative References

[IPFIX-AS] Zseby, T., Boschi, E., Brownlee, N., and B. Claise, "IPFIX Applicability", Work in Progress, July 2007.

[IPFIX-ARCH] Sadasivan, G., Brownlee, N., Claise, B., and J. Quittek, "Architecture for IP Flow Information Export", Work in Progress, September 2006.

[IPFIX-REDUCING] Boschi, E., Mark, L., and B. Claise, "Reducing Redundancy in IP Flow Information Export (IPFIX) and Packet Sampling (PSAMP) Reports", Work in Progress, May 2007.

[PSAMP-PROTO] Claise, B., Quittek, J., and A. Johnson, "Packet Sampling (PSAMP) Protocol Specifications", Work in Progress, December 2007.

[TUEXEN] Tuexen, M. and E. Rescorla, "Datagram Transport Layer Security for Stream Control Transmission Protocol", Work in Progress, November 2007.

[TSVWG-UDP] Eggert, L. and G. Fairhurst, "UDP Usage Guidelines for Application Designers", Work in Progress, February 2008.

[RFC1305] Mills, D., "Network Time Protocol (Version 3) Specification, Implementation and Analysis", RFC 1305, March 1992.

[RFC1924] Elz, R., "A Compact Representation of IPv6 Addresses", RFC 1924, April 1996.

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.

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

[RFC3280] Housley, R., Polk, W., Ford, W., and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3280, April 2002.

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

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

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

[RFC4346] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, April 2006.

[RFC4347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security", RFC 4347, April 2006.

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

[OPENSSL] OpenSSL, "OpenSSL: The Open Source toolkit for SSL/TLS", <http://www.openssl.org/>.

[PEN] IANA, "PRIVATE ENTERPRISE NUMBERS", <http://www.iana.org/assignments/enterprise-numbers>.

## Authors' Addresses

Elisa Boschi
Hitachi Europe
c/o ETH Zurich
Gloriastr. 35
8092 Zurich
Switzerland

Phone: +41 44 6327057
EMail: elisa.boschi@hitachi-eu.com

Lutz Mark
Fraunhofer FOKUS
Kaiserin Augusta Allee 31
10589 Berlin
Germany

Phone: +49 421 2246-206
EMail: lutz.mark@ifam.fraunhofer.de

Juergen Quittek
NEC Europe Ltd.
Kurfuersten-Anlage 36
69115 Heidelberg
Germany

Phone: +49 6221 4342-115
EMail: quittek@nw.neclab.eu

Martin Stiemerling
NEC Europe Ltd.
Kurfuersten-Anlage 36
69115 Heidelberg
Germany

Phone: +49 6221 4342-113
EMail: stiemerling@nw.neclab.eu

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

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

## Full Copyright Statement



## Intellectual Property