

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

P. Aitken
Cisco Systems
February 15, 2013

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

Abstract

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

Status of this Memo

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

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

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

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

Copyright Notice

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

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

Conventions used in this document

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

Table of Contents

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

1 Introduction

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

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

Several scenarios (use cases) can be envisaged:

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

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

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

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

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

2 Terminology

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

3 Method

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

3.1 Equivalence Message Format

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

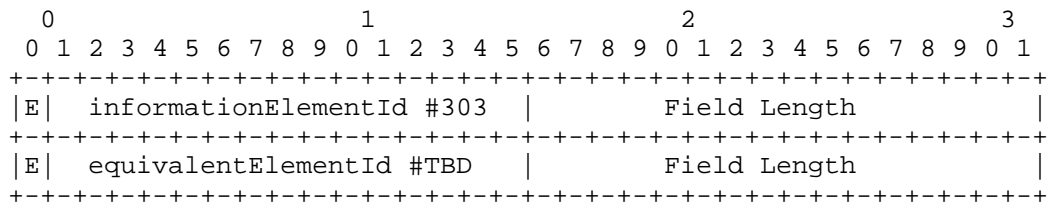


Figure 1: Template for Equivalence Message

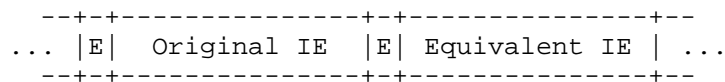


Figure 2: Equivalence Message Data Record

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

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

When the Original Information Element is Enterprise-Specific,

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

```

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

```

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

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

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

```

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

```

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

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

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

```

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

```

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

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

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

4 The Collecting Process's Side

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

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

5 Security Considerations

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

6 IANA Considerations

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

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

Abstract Data Type: Unsigned16

Data Type Semantics: identifier

ElementId: TBD

Status: current

Reference: [this document]

7 References

7.1 Normative References

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

7.2 Informative References

8 Acknowledgements

Thanks to you, dear reader.

9 Author's Addresses

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

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

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: January 05, 2014

T. Eckert, Ed.
R. Penno
A. Choukir
C. Eckel
Cisco Systems, Inc.
July 04, 2013

A Framework for Signaling Flow Characteristics between Applications and
the Network
draft-eckert-intarea-flow-metadata-framework-00

Abstract

This document provides a framework for communicating information elements (a.k.a. metadata) in a consistent manner between applications and the network to provide better visibility of application flows, thereby enabling differentiated treatment of those flows. These information elements can be conveyed using various signaling protocols, including PCP, RSVP, and STUN.

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 January 05, 2014.

Copyright Notice

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

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

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Requirements Language	3
3. Background	3
3.1. Deep packet inspection	4
3.1.1. Benefits	4
3.1.2. Limitation	4
3.2. Explicit signaling methods	5
4. Proposed framework	6
4.1. Overview	7
4.1.1. Common, application independent, IPFIX registered, information elements	7
4.1.2. Cross-protocol information element encoding rules . .	7
4.1.3. Anticipated Usage Models	8
4.1.3.1. Informational	8
4.1.3.2. Advisory	8
4.1.3.3. Service Request	9
4.1.4. Considerations for signaling of common information elements	9
4.1.4.1. Proxy originated information	9
4.1.4.2. Authentication	9
4.1.4.3. Common encoding	10
4.1.4.4. Usage Model to Protocol integration	10
4.2. Proposed common information elements	11
4.2.1. Bandwidth Attributes	12
4.2.1.1. Maximum Bandwidth	12
4.2.1.2. Minimum Bandwidth	12
4.2.1.3. Bandwidth Pool	12
4.2.2. Traffic Class Attributes	12
4.2.2.1. RFC4594-DSCP	12
4.2.2.2. Traffic Class Label (TCL)	12
4.2.3. Acceptable Path Attributes	13
4.2.3.1. Delay Tolerance	13
4.2.3.2. Loss Tolerance	13
4.2.4. Application Identification	14
4.2.4.1. RFC 6759 style application identification	14
4.2.4.2. URL style application identification	14
5. Acknowledgements	16
6. References	16
6.1. Normative References	16
6.2. Informative References	16
Authors' Addresses	17

1. Introduction

This document provides a framework for communicating information elements (a.k.a. metadata) in a consistent manner between applications and the network to provide better visibility of application flows, thereby enabling differentiated treatment of those flows. These information elements can be conveyed using various signaling protocols, including PCP, RSVP, and STUN.

The framework is built around the definition of four key components:

1. A set of application independent information elements (IEs)
2. An encoding of these IEs that is independent of the signaling protocol used as transport
3. Usages of these IEs to support various transactional semantics
4. A mapping of one or more of these usages to an initial set of signaling protocols, including PCP, RSVP, and STUN

This document defines an initial set of IEs, a set of encoding rules, and initial usage model. The actual encoding is defined in [ID-FMD-ENCODE]. Additional documents define the mapping to specific signaling protocols (e.g. RSVP [ID-FMD-RSVP], STUN [I-D.martinsen-mmusic-malice], and PCP [I-D.wing-pcp-flowdata])

2. Requirements Language

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

3. Background

This section provides background on the motivation for the framework.

Identification and treatment of application flows are critical for the successful deployment and operation of applications based on a wide range of signaling protocols. Historically, this functionality has been accomplished to the extent possible using heuristics, which inspect and infer flow characteristics.

Heuristics may be based on port ranges, IP subnetting, or deep packet inspection (DPI), e.g. application level gateway (ALG). Port based solutions suffer from port overloading and inconsistent port usage. IP subnetting solutions are error prone and result in network management hassle. DPI is computationally expensive and becomes a

challenge with the wider adoption of encrypted signaling and secured traffic. An additional drawback of DPI is that the resulting insights are not available, or need to be recomputed, at network nodes further down the application flow path.

The proposed solution allows applications to explicitly signal their flow characteristics to the network. It also provides network nodes with visibility of the application flow characteristics and enables them to contribute to the flow description. The resulting flow description may be communicated as feedback from the network to applications.

3.1. Deep packet inspection

3.1.1. Benefits

Deep Packet Inspection (DPI) and other traffic observation methods (such as performance monitoring) are successfully being used for two type of workflows:

1. Provide network operators with visibility into traffic for troubleshooting, capacity planning, accounting and billing and other off network workflows. This is done by exporting observed traffic analysis via protocol such as IPFIX and SNMP.
2. Provide differentiated network services for the traffic according to network operator defined rule sets, including policing and shaping of traffic, providing admission control, impacting routing, permitting passage of traffic (e.g. firewall functions), etc.

Note: For the context of this document, we consider that DPI starts as early into packets as using ACLs with UDP/TCP port numbers to classify traffic.

3.1.2. Limitation

These two workflows, visibility and differentiated network services, are critical in many networks. However, their reliance on inspection and observation limits the ability to enable these workflows more widely.

- o Simple observation based classification, especially ones relying on TCP/UDP, ports often result in incorrect results due to port overloading (i.e. ports used by applications other than those claiming the port with IANA).

- o More and more traffic becomes encrypted, rendering deep packet inspection impossible or much more complex (e.g. needing to share encryption keys with network equipment).
- o Observation often needs to inspect the control and signaling traffic of applications. This traffic can flow through a different network path than the actual application data traffic. Impacting the traffic behavior is ineffective in those scenarios.
- o Observation of control, signaling and data traffic with DPI will in general result in less insight into the applications intent than if the application was explicitly signaling its intent to the network.
- o Without explicit desire by the application to signal its intent to the network, it will also not consider to explicitly provide authentication to the network. DPI mechanism have a more difficult job in analyzing application traffic when authentication mechanisms are in use (if they even can)
- o Without explicit involvement of the application, network services leveraging DPI traffic classification impact the application behavior by impacting its traffic, but cannot provide explicit feedback to the application in the form of signaling.

3.2. Explicit signaling methods

There are a variety of existing and evolving signaling options that can provide explicit application to network signaling and serve the visibility and differentiated network services workflows where DPI is currently being used. It seems clear that there will be no single one-protocol-fits-all solution. Every protocol is currently defined in its own silo, creating duplicate or inconsistent information models. This results in duplicate work, more operational complexity and an inability to easily convert information between protocols to easily leverage the best protocol option for each specific use case. Examples of existing signaling options include the following:

- o RSVP is the original on path signaling protocol standardized by the IETF. It operates on path out-of-band and could support any transport protocol traffic (it currently supports TCP and UDP). Its original goal was to provide admission control. It gained only limited success with that service. Arguably, its success was impacted by its reliance on router-alert because this often leads to RSVP packets being filtered by intervening networks. To date, more lightweight signaling workflows utilizing RSVP have not been standardized within the IETF.

- o NSIS (next Steps in Signaling) is the next iteration of RSVP-like signaling defined by the IETF. Because it focused on the same fundamental workflow as RSVP admission control as its main driver, and because it did not provide significant enough use-case benefits over RSVP, it has seen even less adoption than RSVP.
- o STUN is an on path, in-band signaling protocol that could easily be extended to provide signaling to on path network devices because it provides an easily inspected packet signature, at least for transport protocols such as UDP and SCTP. Through its extensions TURN and ICE, it is becoming quite popular in application signaling driven by the initial use-case of automatically opening up firewall pinholes and determining the best local and remote addresses for peer-to-peer connectivity (ICE).
- o PCP is a protocol designed to support use cases similar to UPnP firewall traversal. It also can easily be extended to provide more generic application to network signaling for traffic flows. Unlike the prior protocols, it is not meant to be used on path end-to-end but rather independently on one "edge" of a traffic flow. It is therefore an attractive alternative (albeit with challenges under path redundancy) because it allows the introduction of application to network signaling without relying on the remote peer. This is specially useful in multi-domain communications.
- o In addition to these, depending on the devices where it is performed, different degrees of DPI may be used to achieve explicit signaling. For example, inspection of HTTP connections is often viable in high-touch network devices. Such inspection may provide explicit signaling if the application purposely keeps or inserts information elements that are meant to be signaled to the network in the clear, or knowingly uses an encryption scheme shared with the network.

Rather than encourage independent, protocol specific solutions to this problem, this document provides a protocol and application independent framework that can be applied in a consistent fashion across the various protocols.

4. Proposed framework

4.1. Overview

The proposed framework includes the following elements:

4.1.1. Common, application independent, IPFIX registered, information elements

An application media flow may be expressed as a set of information elements that are defined and registered like observation-based IPFIX attributes. We propose leveraging IPFIX as the information model (not necessarily as the transport signaling) for the following reasons:

- o As outlined above, export of traffic information is one of the two big workflows. IPFIX is arguably the most flexible, extensible and best defined option for this. Leveraging the same information model for flow characteristics facilitates export of this information via IPFIX.
- o IPFIX allows for IETF/IANA standardized information elements, but also for unambiguous vendor-defined attributes by including the so-called PEN (Private Enterprise Number) into the information element type. Note that IPFIX has ongoing work to better disseminate vendor specific registration of attributes. The framework defined here expects to be able to leverage the output of that work.

4.1.2. Cross-protocol information element encoding rules

The majority of the protocols listed previously (RSVP, NSIS, STUN/ICE, PCP) require (or favor) compact binary encoding of information elements. This is natively supported by the information element registration of IPFIX.

The IPFIX registry defines each information element's data-type, and there is a native binary network encoding for each of these types. At a minimum, every protocol leveraging common information elements would need to use an encoding that identifies the information element's PEN and IE-ID, and that leverages network standard binary encoding of the value including the length of the value. Including the length of the value into the encoding is required for extensibility because otherwise new information elements could not be introduced without first having all network devices know the data-type, and therefore the length, of the information element. Leveraging network standard binary encoding is equally important to permit network elements to propagate information elements from one protocol to another protocol without understanding the information elements data-type.

In protocols that are not constrained to binary encoding, it is nevertheless highly desirable to include the equivalent information and therefore permit propagation between binary and non-binary transport of information elements without having to understand all information elements.

4.1.3. Anticipated Usage Models

The signaling of information elements may be from application to the network or from network to application. When signaled within a given protocol, the information elements may be interpreted independently of that protocol, or it may be used in combination with the given protocol.

4.1.3.1. Informational

The most simplistic usage model is one in which applications signal information elements describing their anticipated or existing flows into the network along the path of those flows without expecting or requiring anything back from the network. Network elements along the flow path may or may not do something with this information.

This "informational" usage model enables network elements along the path to support the workflows traditionally performed via DPI mechanisms, as described previously.

4.1.3.2. Advisory

This usage model extends the "informational" usage in that the application expects or requests some information back from the network. With this usage, the same information elements apply and may be communicated by the application into the network, but the application indicates its interest in receiving some feedback.

Default values are defined for each information element to unambiguously support cases in which an application does not have a valid value to communicate with the network; rather, it wants the network to provide a value back to it in response. In essence, this allows an application to ask a question and receive an answer from the network. Of course, a network element may provide similar feedback for cases in which an application communicated a non default value as well. Network elements may also provide unsolicited advisory feedback

In all cases, applications are not guaranteed to receive an answer or any specific service from the network. In the event an answer is provided, that answer is similarly not a guarantee of any specific service or treatment by the network. It is to be interpreted as advisory only.

As mentioned previously, the same information elements are used in the signaling from the application to the network as well as from the network to the application. The underlying transport protocol used to carry the information elements is expected to provide the necessary request/response semantics or some other mechanism by which the communication in both directions can be tied together.

4.1.3.3. Service Request

This usage model extends the "advisory" usage to operate as an explicit service request. Unlike the advisory usage, information elements signalled by the application are interpreted by network elements within the context of a service request, and information elements signalled by the network back to the application are interpreted within the context of a response to that request.

As with the advisory usage, the same information elements are used in the signaling from the application to the network as well as from the network to the application. The underlying transport protocol used to carry the information elements is expected to provide the necessary service request/response semantics.

4.1.4. Considerations for signaling of common information elements

4.1.4.1. Proxy originated information

The goal of this framework is to enable applications to explicitly signal common information elements about their traffic flows and optionally receive common information elements from the network as feedback. Nevertheless, it is clear that broad adoption of such technology is improved by enabling the use of proxies. The proxies can provide or amend the flow description information in the absence of Flow Metadata support by the application itself.

4.1.4.2. Authentication

Common information elements should provide for cryptographic authentication by the sender. In general the authentication provides some form of identification of the sender and proves that the common information elements covered by the authentication were originated from, or approved by, that identity.

4.1.4.3. Common encoding

A companion document [ID-FMD-ENCODE] covers recommended encoding rules that take the following aspects into account:

- o Compact binary encoding rules
- o Signaling for both sent and received traffic flows
- o Signaling of standard and vendor specific information elements
- o Minimizes protocol specific definition required to add informational or advisory common information elements into existing transactions
- o Signaling of feedback from the network
- o Identification of originator to support proxies and facilitate mitigation between common information elements from different originators
- o Signaling of authenticators

4.1.4.4. Usage Model to Protocol integration

There is a range of options for how this framework is integrated with a particular transport protocol. We describe two examples we consider useful:

4.1.4.4.1. Common transport informative integration

1. A transport protocol signaling method is defined to carry the common encoded information elements at least in signaling from application to network.
2. If the transport by itself does not already have a mechanism to indicate a purely informative protocol transaction, then a protocol specific indication for this is added.

In result, this integration achieves two option:

1. Informative common information elements can be sent from application to network by using the protocol's method to indicate the purely informational protocol transaction. This option effectively leverages the protocol as transport for additional informative attribute based services without impacting the services and transactions of the protocol otherwise.

2. Informative common information elements can be sent alongside an existing protocol transaction. In this case they may either be ships-in-the-night (triggering informative attribute based services), or they may additionally be used by the policy rules of the protocol transaction itself which could be advisory or service request. All feedback of the transaction would still rely on protocol specific information element (common information elements only used from host to network).

This integration is for example defined in [I-D.wing-pcp-flowdata], [ID-FMD-RSVP], and [I-D.martinsen-mmusic-malice].

4.1.4.4.2. Common transport advisory integration

In addition to the common transport informative integration, the transport encoding is extended to carry the common transport information element in feedback messages from the network to the host /application. The method to indicate informative only transaction, when sending to the network is used to indicate advisory only transaction when signaling from the network.

This option primarily enables informative and advisory usage models, but it can equally interact with pre-existing service-request options of the transport protocol and impact advisory feedback or the service request itself based on that interaction.

4.2. Proposed common information elements

The section defines an initial set of common information elements. These information elements are intended to be added to the set of IANA standardized information elements either by this or associated documents. Additional documents are expected to define additional attributes that can use either IANA or other vendor-PEN.

All information element definition must include the following:

1. Default value to be provided by an application when it does not have an informative value to provide to the network, but is interested in receiving an advisory value of the attribute from the network. If no advisory feedback is requested, and no informative value is known, the attribute may simply not be sent.
2. Conflict resolution in the presence of different values for the same information element (e.g. two peers signaling information elements for both the upstream and downstream direction of a flow include different values for the information element)

4.2.1. Bandwidth Attributes

4.2.1.1. Maximum Bandwidth

This attribute is used to convey the maximum sustained bandwidth for the flow. It is a 64 bit value and is specified in bits per second.

Default Value: 0

Conflict Resolution: Minimum for the set of non default values

4.2.1.2. Minimum Bandwidth

This attribute is used to convey the minimum sustained bandwidth for the flow. It is a 64 bit value and is specified in bits per second. Not sending the Minimum Bandwidth is equivalent to sending the same value as for Maximum Bandwidth.

Default Value: 0

Conflict Resolution: Minimum of the set of non default values

4.2.1.3. Bandwidth Pool

This attribute is used to convey that the traffic dynamically shares bandwidth with other traffic using the same Bandwidth Pool. Variable length GUID (Global Unique ID) of at least 48 bits. The Maximum Bandwidth used by the pool is the largest Maximum Bandwidth indicated by any member, the Minimum Bandwidth of the Pool is the largest Minimum Bandwidth indicated by any member.

4.2.2. Traffic Class Attributes

4.2.2.1. RFC4594-DSCP

This attribute is used to convey the DSCP value appropriate for the flow. It is an 8 bit value. Values signaled are assumed to be in compliance with [RFC4594] or backward compatible extensions thereof. Other values are undefined.

Default Value: 0xff

Conflict Resolution: tbd

4.2.2.2. Traffic Class Label (TCL)

The data type of this information element is a string. It carries the Traffic Class Label defined in

[I-D.ietf-mmusic-traffic-class-for-sdp]. Depending on the outcome of that drafts standardization, the version carried as an information element may be slightly expanded over the its definition for SDP. The TCL is a structured string of the form:

```
<category>.<application>(.adjective)(.adjective)
```

category and application provide a base categorization of the traffic class that attempts to provide a simplified and extensible, framework for the traffic class definitions in [RFC4594]. These base classifications can be refined with zero or more adjectives. Examples of a TCL is "conversational.video.avconf".

Default Value: Empty string

Conflict Resolution: tbd

4.2.3. Acceptable Path Attributes

4.2.3.1. Delay Tolerance

This attribute is used to convey the delay tolerance of an application with respect to the associated flow. When provided by a network element, it indicates the delay tolerance expected of the application with respect to the associated flow. It is a 16 bit field defined in terms of milliseconds.

Default Value: 0

Conflict Resolution: For application to network, the minimum of the set of non default values. For network to application, the maximum of the set of non default values.

4.2.3.2. Loss Tolerance

This attribute is used to convey the loss tolerance of an application with respect to the associated flow. When provided by a network element, it indicates the loss tolerance expected of the application with respect to the associated flow. It is a 16 bit field defined in terms of hundredths of a percent of dropped packets (e.g. 5 == 0.05%, 150 == 1.50%, etc.)

Default Value: 0

Conflict Resolution: For application to network, the minimum of the set of non default values. For network to application, the maximum of the set of non default values.

4.2.4. Application Identification

Application identification is clearly one of the more difficult classification goals. The proposals included here are as of yet not widely vetted:

4.2.4.1. RFC 6759 style application identification

[RFC6759] defines the IPFIX IE-IDs that permit both IANA and vendor specific application identification. Though defined for observation (a.k.a.: DPI), it could also be used with explicit signaling from applications.

Applications that use one of the protocols for which there is an IANA port allocation could explicitly indicate this port via the IANA-L4 engine-id in their application to network signaling. This would identify the application even if the application is not using the IANA assigned port for it. This covers cases in which applications use ports other than registered, such as HTTP servers running on other than 80, or when ports get mapped due to PAT.

To avoid collision with DPI exported IANA-L4 classification, it is necessary to assign a new engine-id for application-self assigned IANA-L4 classification (e.g. new engine-id for IANA-L4-SELF-ASSIGNED). If an application vendor has a PEN, the application can use a PANA-L7-PEN classification with the PEN of the originating application vendor. Likewise, if applications are in general made available via "market" type reseller mechanism (common in mobile device applications), then the application vendor could request an application identification from the market owner and leverage the market owners PEN.

4.2.4.2. URL style application identification

One problem with [RFC6759] style application identification especially non-IANA registered ones is the complexity in making all network elements learn the semantic of the numeric encoding of e.g. the PANA-L7-PEN information element in signaling protocols that only use the numeric encoding of information elements. The second problem may be to determine what PEN to use, because not every developer of an application may be a company that has a PEN or otherwise would intend to apply for one. Application identification via a URL encoded string information element is a way to overcome both issues. Today, almost all applications have some DNS domain associated with them through which they are being marketed or that belongs to the company developing the application. Therefore, one simple form of self assigned application identification is a new IPFIX information element: `UrlAppId`. The value of this information element is an abbreviated URL of the following form:

```
<fqdn> / <app-name> /[ <version> | <other-details> ]
```

The idea is that the owner of `<fqdn>` (fully qualified domain name) is assigning an `<app-name>`, and by signaling both `<domain-name>` and `<app-name>`, this information element provides a self-identifying, unambiguous application identification.

Example:

```
example.com/network-lemmings/sdn-edition
```

A game publishing house or application market operator with the domain name `example.com` is initially allocating the `UrlAppId` `example.com/network-lemmings` to that application. After 35 years, a new variant of the game is released, the SDN edition, and the app-developer decides that it would best like to distinguish this application variant by the above `UrlAppId` `example.com/network-lemmings/sdn-edition`.

In general, different traffic flows within a single application should best not be distinguished via the `UrlAppId`, but instead rely on attributes more specifically targeted for that purpose (such as the `TrafficClassLabel`). If there is no adequate better attribute defined, application developers may choose to use the `other-details` section of the `UrlAppId` to distinguish flows within the same application.

Formally, the only requirement against the `UrlAppId` is that the `fqdn` part is a DNS domain owned by the assigner, and that the rest of the string after the first `/` is as self explanatory as possible.

It should be noted that in the context of DPI, classification of web-based application traffic is very often performed by URL inspection of HTTP traffic. This proposed intent based information element leverages that model and makes it usable where it can not be currently used with just DPI: encrypted HTTP, non-HTTP applications, HTTP applications with non-descriptive URLs, etc.

5. Acknowledgements

The authors would like to thank Dan Wing and Anca Zamfir for their valuable contributions to this document.

6. References

6.1. Normative References

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

6.2. Informative References

- [I-D.ietf-mmusic-traffic-class-for-sdp]
Polk, J., Dhesikan, S., and P. Jones, "The Session Description Protocol (SDP) 'trafficclass' Attribute", draft-ietf-mmusic-traffic-class-for-sdp-03 (work in progress), February 2013.
- [I-D.martinsen-mmusic-malice]
Penno, R., Martinsen, P., Wing, D., and A. Zamfir, "Meta-data Attribute signaling with ICE", draft-martinsen-mmusic-malice-00 (work in progress), July 2013.
- [I-D.wing-pcp-flowdata]
Wing, D., Penno, R., and T. Reddy, "PCP Flowdata Option", draft-wing-pcp-flowdata-00 (work in progress), July 2013.
- [ID-FMD-ENCODE]
Choukir, A., "Protocol Independent Encoding for Signaling Flow Characteristics", 2013, <<http://www.ietf.org>>.
- [ID-FMD-RSVP]
Zamfir, A., "Signaling Flow Characteristics in RSVP", 2013, <<http://www.ietf.org>>.
- [RFC4594] Babiarz, J., Chan, K., and F. Baker, "Configuration Guidelines for DiffServ Service Classes", RFC 4594, August 2006.

[RFC6759] Claise, B., Aitken, P., and N. Ben-Dvora, "Cisco Systems Export of Application Information in IP Flow Information Export (IPFIX)", RFC 6759, November 2012.

Authors' Addresses

Toerless Eckert (editor)
Cisco Systems, Inc.
San Jose
US

Email: eckert@cisco.com

Reinaldo Penno
Cisco Systems, Inc.
170 West Tasman Drive
San Jose 95134
USA

Email: repenno@cisco.com

Amine Choukir
Cisco Systems, Inc.
Lausanne
CH

Email: amchouki@cisco.com

Charles Eckel
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134
US

Email: eckelcu@cisco.com

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

B. Claise
Cisco Systems, Inc.
A. Kobayashi
NTT
B. Trammell
ETH Zurich
June 27, 2013

Operation of the IP Flow Information Export (IPFIX) Protocol on IPFIX
Mediators
draft-ietf-ipfix-mediation-protocol-05.txt

Abstract

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

Status of This Memo

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

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

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

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

Copyright Notice

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

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

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

Table of Contents

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

1. Introduction

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

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

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

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

1.1. IPFIX Documents Overview

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

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

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

IPFIX has a formal description of IPFIX Information Elements, their name, type and additional semantic information, as specified in the IPFIX Information Model [I-D.ietf-ipfix-information-model-rfc5102bis]. The IPFIX Information Element registry [iana-ipfix-assignments] registry is maintained by

IANA. New Information Element definitions can be added to this registry subject to an Expert Review [RFC5226], with additional process considerations described in [I-D.ietf-ipfix-ie-doctors]; that document also provides guidelines for authors and reviewers of new Information Element definitions. The inline export of the Information Element type information is specified in [RFC5610].

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

1.2. IPFIX Mediator Documents Overview

The "IPFIX Mediation: Problem Statement" [RFC5982] provides an overview of the applicability of IPFIX Mediators, and defines requirements for IPFIX Mediators in general terms. This document is of use largely to define the problems to be solved through the deployment of IPFIX Mediators, and to provide scope to the role of IPFIX Mediators within an IPFIX collection infrastructure.

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

Documents specifying the operations of specific Intermediate Processes cover the operation of these Processes within the IPFIX Mediator framework, and comply with the specifications given in this document; they may additionally specify the operation of the process independently, outside the context of an IPFIX Mediator, when this is appropriate. The details of specific Intermediate Processes, when these have additional export specifications (e.g., metadata about the intermediate processing conveyed through IPFIX Options Templates), are each treated in their own document. As of today, these documents are:

1. "IP Flow Anonymization Support", [RFC6235], which describes Anonymization techniques for IP flow data and the export of Anonymized data using the IPFIX protocol.
2. "Flow Selection Techniques" [I-D.ietf-ipfix-flow-selection-tech], which describes the process of selecting a subset of Flows from all Flows observed at an Observation Point, the flow selection motivations, and some specific flow selection techniques.

3. "Exporting Aggregated Flow Data using IP Flow Information Export" [I-D.ietf-ipfix-a9n] which describes Aggregated Flow export within the framework of IPFIX Mediators and defines an interoperable, implementation-independent method for Aggregated Flow export.

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

1.3. Relationship with the IPFIX and PSAMP Protocols

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

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

2. Terminology

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

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

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

The IPFIX Mediation-specific terminology used in this document is defined in "IPFIX Mediation: Problem Statement" [RFC5982], and reused in "IPFIX Mediation: Framework" [RFC6183]. However, since both of

those documents are an informational RFCs, the definitions have been reproduced here along with additional definitions.

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

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

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

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

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

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

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

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

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

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

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

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

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

Specific Intermediate Processes are described below.

Intermediate Conversion Process (as in [RFC6183]): An Intermediate Conversion Process is an Intermediate Process that transforms non-IPFIX into IPFIX or manages the relation among Templates and states of incoming/outgoing transport sessions in the case of transport protocol conversion (e.g., from UDP to SCTP).

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

Intermediate Correlation Process (as in [RFC6183]): An Intermediate Correlation Process is an Intermediate Process that adds information to records, noting correlations among them, or generates new records with correlated data from multiple records (e.g., the production of bidirectional flow records from unidirectional flow records).

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

Intermediate Selection Process (as in [RFC6183]): An Intermediate Selection Process is an Intermediate Process that selects records from a sequence based upon criteria-evaluated record values and passes only those records that match the criteria (e.g., Filtering only records from a given network to a given Collector).

Intermediate Flow Selection Process (as in [I-D.ietf-ipfix-flow-selection-tech]): An Intermediate Flow Selection Process is an Intermediate Process as in [RFC6183] that takes Flow Records as its input and selects a subset of this set as its output.

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

Note: for more information on the difference between Intermediate Flow Selection Process and Intermediate Selection Process, see Section 4 in [I-D.ietf-ipfix-flow-selection-tech].

3. Handling IPFIX Message Headers

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

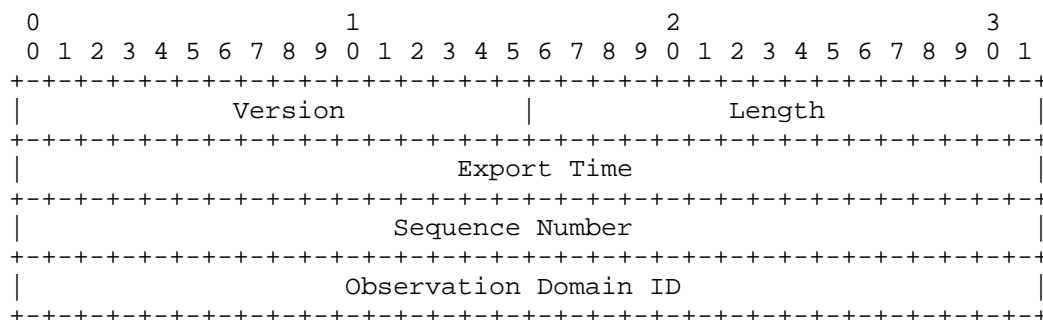


Figure 1: IP Message Header format

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

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

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

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

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

Observation Domain ID: A 32-bit identifier of the Observation Domain that is locally unique to the Exporting Process. The Exporting Process uses the Observation Domain ID to uniquely identify to the Collecting Process the Observation Domain that metered the Flows. It is RECOMMENDED that this identifier also be unique per IPFIX Device. Collecting Processes SHOULD use the Transport Session and the Observation Domain ID field to separate different export streams originating from the same Exporter. The Observation Domain ID SHOULD be 0 when no specific Observation Domain ID is relevant for the entire IPFIX Message, for example, when exporting the Exporting Process Statistics, or in case of a hierarchy of Collectors when aggregated Data Records are exported. See Section 4.1 for special considerations for Observation Domain management while passing unmodified templates through an IPFIX Mediator, and Section 5 for guidelines for preservation of original Observation Domain information at an IPFIX Mediator.

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

If an IPFIX Mediator receives an IPFIX Message composed of Template Withdrawals and Template Sets, and if the IPFIX Mediator forwards

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

4. Template Management

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

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

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

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

4.1. Passing Unmodified Templates through an IPFIX Mediator

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

As specified in [I-D.ietf-ipfix-protocol-rfc5101bis], the Template IDs are unique per Exporter, per Transport Session, and per Observation Domain. As there is no guarantee that, for similar Template Records, the Template IDs received on the incoming Transport Session and exported to the outgoing Transport Session would be same,

the IPFIX Mediator MUST maintain a Template Mapping composed of related received and exported (Options) Template Records:

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

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

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

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

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

For example, Figure 2 displays an example of an Intermediate Flow Selection Process, re-distributing Data Records to Collectors on the basis of customer networks, i.e. the Route Distinguisher (RD). In this example, the Template Record received from the Exporter #1 is reused towards Collector #1, Collector #2, and Collector #3, for the customer #1, customer #2, and customer #3, respectively. In this example, the outgoing Template Records exported to the different Collectors are identical. As a reminder that the Template ID uniqueness is local to the Transport Session and Observation Domain

that generated the Template ID, a mix of Template ID 256 and 257 has been used.

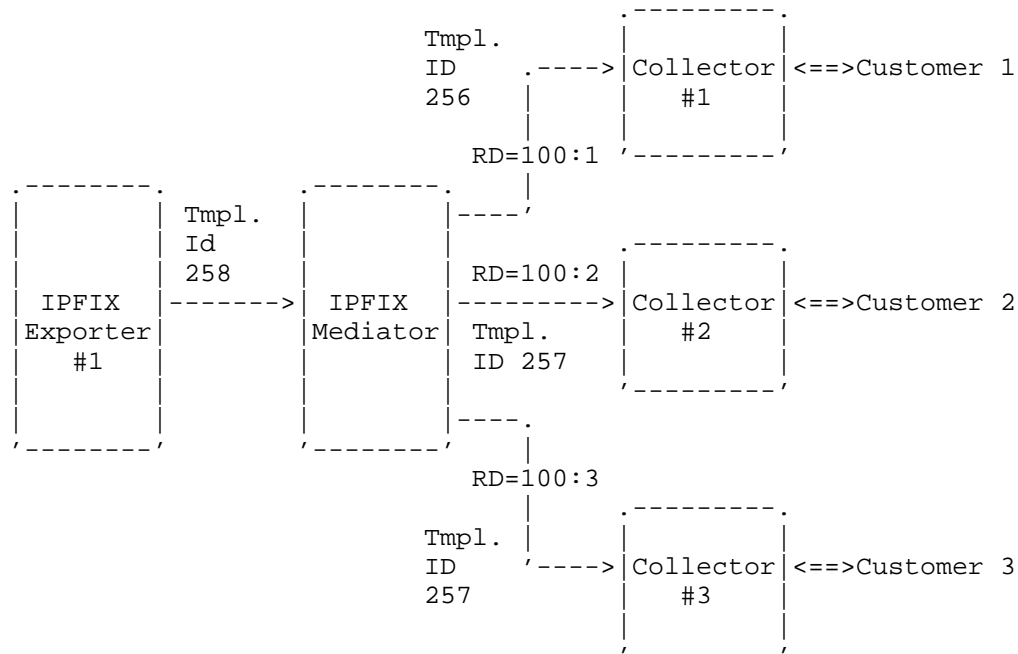


Figure 2: Intermediate Flow Selection Process example

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

Template Entry A:

Incoming Transport Session Information (from Exporter#1):

Source IP: <Exporter#1 export IP address>
 Destination IP: <IPFIX Mediator IP address>
 Protocol: SCTP
 Source Port: <source port>
 Destination Port: 4739 (IPFIX)

Observation Domain Id: <Observation Domain ID>

Template Id: 258

Metadata scoped to the Template : <not applicable in this case>

Template Entry B:

Outgoing Transport Session Information (to Collector#1):

Source IP: <IPFIX Mediator IP address>
 Destination IP: <IPFIX Collector#1 IP address>
 Protocol: SCTP
 Source Port: <source port>

Destination Port: 4739 (IPFIX)
 Observation Domain Id: <Observation Domain ID>
 Template Id: 256
 Metadata scoped to the Template : <not applicable in this case>

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

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

Figure 3: Template Mapping example: templates

The Template Mapping corresponding to figure 3 is displayed in figure 4:

```

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

```

Figure 4: Template Mapping example: mappings

Alternatively, the Template Mapping may be optimized as in figure 5:

```

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

```

Figure 5: Template Mapping example2: mappings

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

4.1.1. Template Mapping and Information Element Ordering

In the situation where Original Exporters each export an (Options) Template to a single IPFIX Mediator, and the (Options) Template Record contains the same Information Elements but in different order, should the IPFIX Mediator maintain a Template Mapping with a single Export Template Record (see figure 6) or should the IPFIX Mediator maintain multiple independent Template Records (see figure 7) before re-exporting to the Collector?

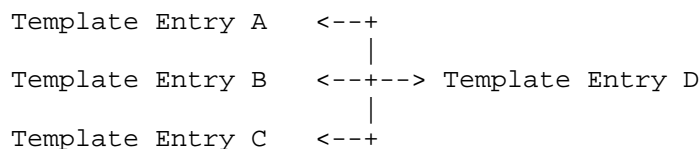


Figure 6: Template Mapping and Ordering: a single Export Template Record

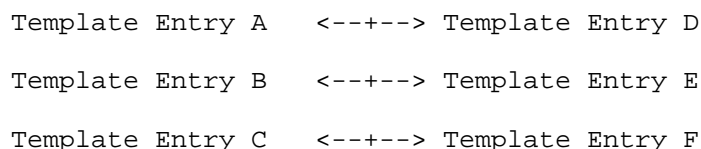


Figure 7: Template Mapping and Ordering: multiple Export Template Records

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

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

while the second scope defines the sampling function, the order of the scope is important. Applying the sampling function first, followed by the filtering function, would lead to potentially different Data Records than applying the filtering function first, followed by the sampling function."

If an IPFIX Mediator receives, from multiple Exporters, Template Records with identical Information Elements, but ordered differently, it SHOULD consider those Template Records as identical, subject to metadata information in the associated Options Template (for example, the Flow Key Options Template. See Section 10.2).

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

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

4.2. Creating New Templates at an IPFIX Mediator

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

In this situation, the IPFIX Mediator doesn't need to maintain a Template Mapping, as it generates its own series of (Options) Template Records. However, the following special case might still require a Template Mapping, i.e. a situation where the IPFIX Mediator, typically containing an Intermediate Conversion Process, Intermediate Aggregation Process, or Intermediate Anonymization Process in case of black-marker Anonymization [RFC6235], generates new (Options) Template Records based on what it receives from the Exporter(s), and based on the Intermediate Process function. In such a case, it's important to keep the correlation between the received (Options) Template Records and derived (Options) Template Records in the Template Mapping. These Template Mappings would be kept as in Section 4.1, except that the exported Template would not be identical to the received Template.

4.3. Handling Unknown Information Elements

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

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

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

5. Preserving Original Observation Point Information

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

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

- o A single Original Exporter (represented by the `originalExporterIPv4Address` or `originalExporterIPv6Address` Information Elements)
- o A list of Original Exporters (represented by a list of `originalExporterIPv4Address` or `originalExporterIPv6Address` Information Elements).

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

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

Any combination of the above representations is possible. An example of an Original Observation Point for an Intermediate Aggregation Process is displayed in figure 8.

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

Figure 8: Complex Observation Point Definition Example

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

The most generic way to export the Original Observation Point is to use a subTemplateMultiList, with the semantic "exactlyOneOf". Taking the previous example, the encoding in figure 9 can be used.

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

Figure 9: Complex Observation Point Definition Example: Templates

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

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

5.1. `originalExporterIPv4Address` Information Element

Name: `originalExporterIPv4Address`
Description: The IPv4 address used by the Exporting Process on an Original Exporter, as seen by the Collecting Process on an IPFIX Mediator. Used to provide information about the Original Observation Points to a downstream Collector.
Data Type: `ipv4Address`
ElementId: TBD1

5.2. `originalExporterIPv6Address` Information Element

Name: `originalExporterIPv6Address`
Description: The IPv6 address used by the Exporting Process on an Original Exporter, as seen by the Collecting Process on an IPFIX Mediator. Used to provide information about the Original Observation Points to a downstream Collector.
Data Type: `ipv6Address`
ElementId: TBD2

6. Managing Observation Domain IDs

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

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

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

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

In some cases where the IPFIX Message Header can't contain a consistent Observation Domain for the entire IPFIX Message, but the Flow Records exported from the IPFIX Mediator should anyway contain the Observation Domain of the Original Exporter, the (Options) Template Record must contain the originalObservationDomainId Information Element, specified in Section 6.1. When an IPFIX Mediator receives Flow Records containing the originalObservationDomainId Information Element, the IPFIX Mediator MUST NOT modify its value(s) when composing new Flow Records with the originalObservationDomainId Information Element.

6.1. originalObservationDomainId Information Element

Name: originalObservationDomainId
Description: The Observation Domain ID reported by the Exporting Process on an Original Exporter, as seen by the Collecting Process on an IPFIX Mediator. Used to provide information about the Original Observation Domain to a downstream Collector.
Data Type: unsigned32
Data Type Semantics: identifier
ElementId: TBD3

7. Timing Considerations

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

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

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

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

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

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

8. Transport Considerations

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

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

9. Collecting Process Considerations

Any Collecting Process compliant with [I-D.ietf-ipfix-protocol-rfc5101bis] can receive IPFIX Messages from an IPFIX Mediator. If the IPFIX Mediator uses IPFIX Structured Data [RFC6313] to export Original Exporter Information as in Section 5, the Collecting Process MUST support [RFC6313].

10. Specific Reporting Requirements

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

Analogous to the Metering Process Reliability Statistics Options Template, also specified in [I-D.ietf-ipfix-protocol-rfc5101bis], Mediators SHOULD implement the Intermediate Process Reliability Statistics Options Template, specified in the Section 10.1.

The Flow Keys Options Template, as specified in [I-D.ietf-ipfix-protocol-rfc5101bis], may require special handling at an IPFIX Mediator as described in Section 10.2.

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

10.1. Intermediate Process Reliability Statistics Template

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

IE	Description
observationDomainId [scope]	An identifier of the Observation Domain (of messages exported by this Mediator), locally unique to the Intermediate Process, to which this statistics record applies.
intermediateProcessId [scope]	An identifier for the Intermediate Process to which this statistics record applies.
ignoredFlowRecordTotalCount	The total number of Data Records received but not processed by the Intermediate Process.
time first record ignored	The timestamp of the first record that was ignored by the Intermediate Process. For Data Records containing timestamp ranges, this SHOULD be taken from the start timestamp of the range; for data records containing no timing information, this SHOULD be taken from the Export Time in the message header of the containing IPFIX Message. For this timestamp, any of the following timestamp can be used: observationTimeSeconds, observationTimeMilliseconds, observationTimeMicroseconds, or observationTimeNanoseconds.
time last record ignored	The timestamp of the last record that was ignored by the Intermediate Process. For Data

	Records containing timestamp ranges, this SHOULD be taken from the end timestamp of the range; for data records containing no timing information, this SHOULD be taken from the Export Time in the message header of the containing IPFIX Message. For this timestamp, any of the following timestamp can be used: observationTimeSeconds, observationTimeMilliseconds, observationTimeMicroseconds, or observationTimeNanoseconds.
--	---

10.2. Flow Key Options Template

The Flow Keys 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. This Options Template is defined in section 4.4 of [I-D.ietf-ipfix-protocol-rfc5101bis], and SHOULD be used by Mediators for export as defined there.

When an Intermediate Process exports Data Records containing different Flow Keys from those received from the Original Exporter, and the Original Exporter sent a Flow Keys Options record to the IPFIX Mediator, the IPFIX Mediator MUST export a Flow Keys Options record defining the new set of Flow Keys.

10.3. intermediateProcessId Information Element

Name: intermediateProcessId

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

Data Type: unsigned32

Data Type Semantics: identifier

ElementId: TBD4

10.4. ignoredFlowRecordTotalCount Information Element

Name: ignoredFlowRecordTotalCount

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

Data Type: unsigned64

Data Type Semantics: totalCounter

ElementId: TBD5

11. Configuration Management

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

12. Security Considerations

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

By design, IPFIX Mediators are "men-in-the-middle": they intercede in the communication between an Original Exporter (or another upstream IPFIX Mediator) and a downstream Collecting Process. This has two important implications for the level of confidentiality provided across an IPFIX Mediator, and the ability to protect data integrity and Original Exporter authenticity across an IPFIX Mediator. These are addressed in more detail in the Security Considerations for IPFIX Mediators in [RFC6183].

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

simply falsify the information about upstream Transport Sessions. In situations where information about the chain of mediation is important, it must be determined out of band.

13. IANA Considerations

This document specifies new IPFIX Information Elements, `originalExporterIPv4Address` in Section 5.1, `originalExporterIPv6Address` in Section 5.2, `originalObservationDomainId` in Section 6.1, `intermediateProcessId` in Section 10.3, and `ignoredFlowRecordTotalCount` in Section 10.4, to be added to the IPFIX Information Element registry [iana-ipfix-assignments]. [IANA NOTE: please add the five Information Elements as specified in the references subsections, change TBD1, TBD2, TBD3, TBD4, and TBD5 in this document to reflect the assigned identifiers, put the Status as current, insert THISRFC into the Reuquester entry, insert 0 for the Revision, and use the current date for Date.]

14. Acknowledgments

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

15. References

15.1. Normative References

- [RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.
- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3758] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", RFC 3758, May 2004.
- [RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.

- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.
- [RFC5655] Trammell, B., Boschi, E., Mark, L., Zseby, T., and A. Wagner, "Specification of the IP Flow Information Export (IPFIX) File Format", RFC 5655, October 2009.
- [RFC6313] Claise, B., Dhandapani, G., Aitken, P., and S. Yates, "Export of Structured Data in IP Flow Information Export (IPFIX)", RFC 6313, July 2011.
- [RFC6615] Dietz, T., Kobayashi, A., Claise, B., and G. Muenz, "Definitions of Managed Objects for IP Flow Information Export", RFC 6615, June 2012.
- [RFC6727] Dietz, T., Claise, B., and J. Quittek, "Definitions of Managed Objects for Packet Sampling", RFC 6727, October 2012.
- [RFC6728] Muenz, G., Claise, B., and P. Aitken, "Configuration Data Model for the IP Flow Information Export (IPFIX) and Packet Sampling (PSAMP) Protocols", RFC 6728, October 2012.
- [I-D.ietf-ipfix-protocol-rfc5101bis]
Claise, B. and B. Trammell, "Specification of the IP Flow Information eXport (IPFIX) Protocol for the Exchange of Flow Information", draft-ietf-ipfix-protocol-rfc5101bis-08 (work in progress), June 2013.
- [I-D.ietf-ipfix-information-model-rfc5102bis]
Claise, B. and B. Trammell, "Information Model for IP Flow Information eXport (IPFIX)", draft-ietf-ipfix-information-model-rfc5102bis-10 (work in progress), February 2013.
- [I-D.ietf-ipfix-flow-selection-tech]
D'Antonio, S., Zseby, T., Henke, C., and L. Peluso, "Flow Selection Techniques", draft-ietf-ipfix-flow-selection-tech-18 (work in progress), May 2013.
- [I-D.ietf-ipfix-ie-doctors]
Trammell, B. and B. Claise, "Guidelines for Authors and Reviewers of IPFIX Information Elements", draft-ietf-ipfix-ie-doctors-07 (work in progress), October 2012.
- [I-D.ietf-ipfix-a9n]

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

15.2. Informative References

- [RFC3917] Quittek, J., Zseby, T., Claise, B., and S. Zander, "Requirements for IP Flow Information Export (IPFIX)", RFC 3917, October 2004.
- [RFC3954] Claise, B., "Cisco Systems NetFlow Services Export Version 9", RFC 3954, October 2004.
- [RFC5470] Sadasivan, G., Brownlee, N., Claise, B., and J. Quittek, "Architecture for IP Flow Information Export", RFC 5470, March 2009.
- [RFC5472] Zseby, T., Boschi, E., Brownlee, N., and B. Claise, "IP Flow Information Export (IPFIX) Applicability", RFC 5472, March 2009.
- [RFC5476] Claise, B., Johnson, A., and J. Quittek, "Packet Sampling (PSAMP) Protocol Specifications", RFC 5476, March 2009.
- [RFC5610] Boschi, E., Trammell, B., Mark, L., and T. Zseby, "Exporting Type Information for IP Flow Information Export (IPFIX) Information Elements", RFC 5610, July 2009.
- [RFC5982] Kobayashi, A. and B. Claise, "IP Flow Information Export (IPFIX) Mediation: Problem Statement", RFC 5982, August 2010.
- [RFC6183] Kobayashi, A., Claise, B., Muenz, G., and K. Ishibashi, "IP Flow Information Export (IPFIX) Mediation: Framework", RFC 6183, April 2011.
- [RFC6235] Boschi, E. and B. Trammell, "IP Flow Anonymization Support", RFC 6235, May 2011.
- [iana-ipfix-assignments] Internet Assigned Numbers Authority, ., "IP Flow Information Export Information Elements (<http://www.iana.org/assignments/ipfix/ipfix.xml>)", .
- [POSIX.1] IEEE, ., "IEEE 1003.1-2008 - IEEE Standard for Information Technology - Portable Operating System Interface", .

Authors' Addresses

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

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

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

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

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

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

IPFIX Working Group
Internet-Draft
Intended Status: Standards Track
Expires May 13, 2013

C. Inacio
Carnegie Mellon University
November 9, 2012

Private Enterprise Information Elements Registry Exchange
<draft-inacio-ipfix-penie-00.txt>

Abstract

This extension to the IPFIX protocol is intended to provide a mechanism for IPFIX exporters which export private information elements to also transmit information to the collectors. The mechanism is designed to be able to send a URI with information about the private information elements via an options template.

Status of this Memo

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

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

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

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

This Internet-Draft will expire in May 2013.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document.

Table of Contents

1. Introduction	4
2. Options Record Format	4
3. Registry Design	5
3.1. Registry Introduction	6
3.2. Registry Informational Elements	6
3.3. Registry Formatting	7
6. IANA Considerations	7
6. References	7
6.1. Normative References	7
6.2. Informative References	8
Authors' Addresses	8
Appendix A.	8
99.0. To be removed	10
99.1. Formatting End of Page	12

1. Introduction

The IPFIX protocol [RFC5101] defined a significant information element set for a large number of relevant network activities. The IPFIX protocol was also designed with the ability to extend its information model both via a standards based mechanism, via an IANA registry, to add elements of general interest, but also the ability to add elements via private enterprise numbers to define elements of limited interest. Beyond adding the ability to pass new information elements, the IPFIX protocol is designed to allow collectors the ability to skip information elements which cannot be comprehended.

IPFIX was extended in RFC 5610 IPFIX Semantic Type Information to allow IPFIX send type semantic information about information elements. This mechanism allows an IPFIX exporter to send type semantic information along with a common name about an information element to the collector. This allows information elements that the collector would otherwise not be able to comprehend to provide much more information.

The mechanism proposed here extends the Semantic Type Information in two ways. First, it allows a more complex definition of information to be presented, capturing the possible relationships contained within the IPFIX Structured Data extension. The IPFIX Structured Data extension, having completed after the Semantic Type Information, is not covered in the Semantic Type Information. Secondly, by moving the information element metadata out from the potentially resource constrained IPFIX data channel, this extension allows a richer and comprehensive set of metadata to be expressed.

2. Options Record Format

The mechanism used to transmit the URI information from the exporter to the collector is an options template with a URI. The URI contains a pointer which provides well formatted, as specified in section 3, information about the semantic type information and description of the private information elements.

										1										2										3									
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1								
Set ID = 3										Length = 14																													
Template ID = 257										Field Count =																													
Scope Field Count =										0 private ent. number										346																			
Field Length = 4										PEN Registry URI										XXX																			

[illegible]

Figure 1: Example PENIE Template Definition

[illegible]

Figure 2: Example PENIE Data Record

The options record allows for creating a URI reference for a private enterprise number. It is important to note that more than one URI per a single private enterprise number. The burden of resolving all declared registries falls onto the collector to be able to decode all information elements received from the exporters.

3. Registry Design

3.1. Registry Introduction

The registry design goals are to capture all the information that IPFIX Type Information [RFC5610] provides about individual elements, but to also present more metadata about both individual elements as well as have the ability to provide more information about a collection of elements.

3.2. Registry Informational Elements

The top level of a registry contains the following additional elements:

- o Registry ID - This is an ID that can be used by the creator of the registry to be able to track the registry as a unique item.
- o Version - Indicates the release version of the registry.
- o Name - A common name that can be used to refer to the registry.
- o Security Type - This is a new entry type that allows the complete set of elements defined in the registry to be contained within a security type class. By allowing the collector to understand the security type, if present, of the information elements a new class of actions may be taken by a collector implementation.
- o Policy Type - Similar to the security type, this new entry allows the complete set of elements defined in the registry to be contained within a policy type class. Again, similar to the security type class, the collector may take new actions based upon understanding the policy type of an information element.
- o Canonical URI - This is the canonical URI to be able to locate the authoritative version of the registry.
- o Root EID - This defines the Private Enterprise ID for all elements defined in the registry.
- o Copyright - Optionally the copyright information for the registry
- o Contact - Contact information to be able to contact the publisher of the registry.
- o Directory - (I can't remember, but its a URI).

Each information element defined in the registry are as follows:

- o ID - The information element ID.
- o PEN - The private enterprise number.
- o Data type - The data type of the element, as defined in RFC 5610.
- o Semantics - The semantic of the element, as defined in RFC 5610.
- o Units - The units of the element, as defined in RFC 5610.
- o Description - The human readable (and hopefully understandable) description of the element.
- o MIME Path - An optional MIME path definition of the element.

3.3. Registry Formatting

XML or JSON or ??? format to be added here.

5. Security Considerations

There are no security considerations relevant to this document, beyond the security considerations necessary in the IPFIX protocol specification [RFC5101] and its successors.

6. IANA Considerations

IANA needs to create two registries with expert review:

Security types Policy types

and create a code point for a new information element.

6. References

6.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC5101] Claise, B., "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information", RFC 5101, January 2008.
- [RFC5102] Quittek, J., Bryant, S., Claise, B., Aitken, P., and Meyer, J., "Information Model for IP Flow Information Export",

RFC 5102, January 2008.

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

[RFC4234] Crocker, D. and P. Overell, "Augmented BNF for Syntax
Specifications: ABNF", RFC 4234, October 2005.

6.2. Informative References

[2223BIS] Reynolds, J. and R. Braden, "Instructions to Request for
Comments (RFC) Authors", draft-rfc-editor-rfc2223bis-
08.txt, August 2004.

Authors' Addresses

Christopher Inacio
Carnegie Mellon University
4500 5th Avenue
Pittsburgh, PA 15213
USA

EMail: inacio@sei.cmu.edu

Appendix A.

Relax-NG based definition of a proposed schema. Can be processed with trang
to create a well defined XML XSD file.

```
namespace r = "http://www.ietf.org/ipfix/ipfix-private-element-registry/1.0"
```

```
# It's unclear what the right way of referencing an info element ought  
# to be. By PEN/ID pair? By some XML ID? Both has problems. Leave it  
# text for now.  
info-elem-ref = text
```

```
r-data-type = element r:data-type {  
  "octetArray" |  
  "unsigned8" |  
  "unsigned16" |  
  "unsigned32" |  
  "unsigned64" |  
  "signed8" |  
  "signed16" |  
  "signed32" |  
  "signed64" |  
  "float32" |  
  "float64" |
```

```
"boolean" |
"macAddress" |
"string" |
"dateTimeSeconds" |
"dateTimeMilliseconds" |
"dateTimeMicroseconds" |
"dateTimeNanoseconds" |
"ipv4Address" |
"ipv6Address" |
"basicList" |
"subTemplateList" |
"subTemplateMultiList"
}

r-semantic = element r:semantic {
  "default" |
  "quantity" |
  "totalCounter" |
  "deltaCounter" |
  "identifier" |
  "flags" |
  "list"
}

r-units = element r:units {
  "none" |
  "bits" |
  "octets" |
  "packets" |
  "flows" |
  "seconds" |
  "milliseconds" |
  "microseconds" |
  "nanoseconds" |
  "4-octet words" |
  "messages" |
  "hops" |
  "entries"
}

#r-value-map = element r:value-map {
#   attribute type {"integer" | "text" },
#   {element value }
#}

r-element = element r:element {
  element r:id { xsd:integer {minInclusive="0" maxInclusive="65535"}} &
  element r:private-enterprise-number { xsd:integer {minInclusive="0" maxInclu
sive="4294967295"}}? &
```

```

    r-data-type &
    r-semantics? &
    r-units? &
    element r:range-begin { text }? &
    element r:range-end { text }? &
    element r:name { xsd:token } &
    element r:description { text } &

    element r:mime-path { text }?

# element r:value-map {
#     attribute type {"integer" | "text"},
# }
}

r-registry = element r:registry {
    element r:id { xsd:anyURI } &
    element r:name { text } &

    # Should these two be required or optional?
    element r:security-type { text } &
    element r:policy-type { text } &

    element r:url { xsd:anyURI } &
    element r:root-eid { xsd:integer } &
    (r-registry | r-element)*
}

r-enterprise-registry = element r:enterprise-registry {
    element r:name { text } &
    element r:copyright { text } &
    element r:contact { text } &
    element r:private-enterprise-number { xsd:integer } &
    element r:directory { xsd:anyURI } &
    r-registry*
}

start = r-enterprise-registry

```

99.0. To be removed

The RFC Editor generally uses the simplest nroff features, basically the "-ms" macro package and the following few basic nroff directives:

DIRECTIVE	FUNCTION
.ce	Center following line.

.ti # 'temporary indent' -- # is number of spaces.
 Indents only the line immediately following.

.in # Change indentation to # spaces

.nf 'No fill': begin block of text to be displayed.

.fi Fill (i.e., left-justify, line wrap)

.ne # 'need' -- Keep following # lines on same page

.bp Break page

.br Break line

.KS 'Keep Start' -- lines up to .KE on same page

.KE 'Keep End' -- end of 'keep' block

Nroff also has a '.sp' (space) directive to insert a blank line. However, it is far easier (and more readable) to use the fact that each blank line in the nroff source creates a blank line in the output.

Nroff includes many variations on the trivial commands shown above. For example, indentation can be specified relative to the current indentation, using '.in +#' or '.in -#'. Authors are welcome to use such features, but for simplicity this template uses only the simplest set of commands.

Some authors who are proficient in nroff will wish to use more advanced features, including perhaps their own macros. This is a private matter for the author, unless and until the document is submitted to the RFC Editor for publication as an RFC. Upon document submission, the RFC Editor will request the nroff source, if any. If the source is sufficiently straightforward, it will be used by the RFC Editor to speed the publication process. If not, the RFC Editor will generate a new nroff source, generally using the simple subset above.

The considerations here are as follows:

- o Defined macros (beyond the -ms package) must be in-line at the front of the source. The RFC Editor is currently prepared to maintain only one source file for each published RFC.
- o Some of the editors are not nroff experts, and even those who may be do not have the time to figure out some complex/obscure

macro. If any special knowledge about these macros is needed to modify the text for editorial purposes, the RFC Editor will find it more expedient to generate a new .nroff source for the document.

- o The RFC Editor does not keep a distinct Make file for each RFC, so it is not helpful to send us a tar file or shar script that magically makes a directory and builds an RFC. Our primary input is a .txt file, with a .nroff file as a possible secondary input. When the RFC is published, the RFC Editor will archive a .txt file and a corresponding \&.nroff file.

In other words, keep it simple and you can help us a lot; don't show off your programming prowess and waste our time.

99.1. Formatting End of Page

The Unix command to create a formatted Internet Draft is:

```
"nroff -ms input-file.nroff > output-file.txt"
```

However, nroff will not follow the RFC standard format for a page: a Form feed (FF or Control-L)) after the last visible line on the page and no extra line feeds before the first visible line of the next page. We want:

```
last visible line on page i
^L
first visible line on page i+1
```

We invented hacks to fix this. The original hack was a "sed" script that called a "C" program called "pg". More recently, we have been using a simple Perl script (see Appendix A). Then the command to process the nroff source file becomes:

```
nroff -ms input-file.nroff | fix.pl > output-file.txt
```

For example:

```
nroff -ms 2-nroff.template | fix.pl > 2-nroff.template.txt
```

IPFIX

Internet Draft

Intended status: Informational

Expires: December 2013

June 16, 2013

R. Krishnan

Brocade Communications

Ning So

Tata Communications

Flow-state dependent packet selection techniques

draft-krishnan-ipfix-flow-aware-packet-sampling-05.txt

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, except to publish it as an RFC and to translate it into languages other than English.

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

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

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>

This Internet-Draft will expire on December 18, 2013.

Copyright Notice

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

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

Abstract

The demands on the networking infrastructure and thus the switch/router bandwidths are growing exponentially; the drivers are bandwidth hungry rich media applications, inter data center communications etc. Using sampling techniques, for a given sampling rate, the amount of samples that need to be processed is increasing exponentially especially for applications like security threat detection. This draft elaborates on flow-state dependent packet selection techniques and the relevant information models. It describes how these techniques can be effectively used to reduce the number of samples for applications like security threat detection.

Table of Contents

1. Introduction.....	2
1.1. Acronyms.....	3
1.2. Terminology.....	3
2. Flow-state dependent packet selection techniques.....	3
2.1. Information Model for flow-state dependent packet selection technique configuration.....	4
2.2. Handling Inactive/Misidentified Large Flows.....	5
2.3. Flow-state dependent packet selection - sample and hold...	6
2.4. IANA Considerations.....	6
2.4.1. Registration of Information Elements.....	6
2.4.1.1. largeFlowObservationInterval.....	6
2.4.1.2. largeFlowBandwidthThreshold.....	6
3. Current sampling techniques for security threat detection.....	7
4. Application of flow-state dependent packet selection techniques for security threat detection.....	7
4.1. Applicability of flow-state dependent packet selection technique suggested in [ESVA].....	Error! Bookmark not defined.
4.2. Applicability of flow-state dependent packet selection technique suggested in [VRM].....	Error! Bookmark not defined.
4.3. Simulation.....	9
5. Security Considerations.....	9
6. Operational Considerations.....	9
7. Acknowledgements.....	9
8. References.....	10
8.1. Normative References.....	10
8.2. Informative References.....	10

1. Introduction

This draft expands on the flow-state dependent packet selection techniques described in [FLSEC] for identifying long-lived large

flows and the relevant information models. This draft also describes a practical use case for efficient behavioral security detection, like Denial of Service (DOS) attacks etc., using flow-state dependent packet selection techniques.

1.1. Acronyms

DOS: Denial of Service

GRE: Generic Routing Encapsulation

MPLS: Multi Protocol Label Switching

NVGRE: Network Virtualization using Generic Routing Encapsulation

TCAM: Ternary Content Addressable Memory

STT: Stateless Transport Tunneling

VXLAN: Virtual Extensible LAN

1.2. Terminology

Large flow(s): long-lived large flow(s)

Small flow(s): long-lived small flow(s) and short-lived small/large flow(s)

2. Flow-state dependent packet selection techniques

Expanding on the work in [FLSEC] and [RFC 5475], this draft suggests additional techniques for flow-state dependent packet selection for identifying large flows. One of these techniques is called Multistage Filters which is described in [ESVA]. This technique helps in automatically identifying large flows with a low false positive rate. This technique can be implemented as an inline solution in switches/routers and would be expected to operate at line rate.

Besides the Multistage filters technique described in [ESVA],

- 1) The technique suggested in [VRM] is also applicable. [VRM] suggests techniques for automatically identifying large flows using rotating conservative counting Bloom filters with periodic decay. This technique has a low false positive rate in large flow misidentification.

- 2) The sample and hold technique suggested in [ESVA] is also applicable. This technique has a low false positive rate in large flow misidentification.

The large flows which are automatically identified using the above techniques are populated in the IPFIX flow cache [RFC 6728]. If a large flow already exists in the IPFIX flow cache, the above techniques are not applied - this is the reason these are called flow-state dependent packet selection techniques.

Please note that there is a finite probability of small flows being misidentified as large flows. These are handled as described in the section 2.2 "Handling Inactive/Misidentified Large Flows".

2.1. Information Model for flow-state dependent packet selection technique configuration

From a bandwidth and time duration perspective, in order to identify large flows we define an observation interval and observe the bandwidth of the flow over that interval. A flow that exceeds a certain minimum bandwidth threshold over that observation interval would be considered a large flow.

The two configuration parameters -- the observation interval, and the minimum bandwidth threshold over that observation interval -- should be programmable in a switch or a router to facilitate handling of different use cases and traffic characteristics are defined below.

largeFlowObservationInterval: The minimum time interval to observe a flow before performing further processing of the flow. Unit is in milliseconds.

`largeFlowBandwidthThreshold`: The minimum bandwidth of the flow during the observation interval for declaring the flow a large flow. Unit is in Mbps.

For example, a flow which is at or above 10 Mbps for a time period of at least 30 seconds could be declared a large flow.

Below is the list of flow-state dependent packet selection technique Information Elements:

-----+		+-----+		+-----+	
-----+		ID	Name	ID	Name
-----+		+-----+		+-----+	
-----+		TBD	largeFlowObservationInterval	TBD	largeFlowBandwidthThreshold
-----+		1		2	
-----+		+-----+		+-----+	
-----+		+-----+		+-----+	

2.2. Handling Inactive/Misidentified Large Flows

Once a flow has been recognized as a large flow, it should continue to be recognized as a large flow as long as the traffic received during an observation interval exceeds some fraction of the bandwidth threshold, for example 80% of the bandwidth threshold. If the traffic received during an observation interval falls below a fraction of the bandwidth threshold, the large flow should be removed from the IPFIX flow cache.

2.3. Flow-state dependent packet selection - sample and hold

[FLSEC] suggests some information model parameters for the sample and hold technique suggested in [ESVA]. The large flow information model parameters suggested in section 2.1 are complementary to these.

2.4. IANA Considerations

2.4.1. Registration of Information Elements

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

IANA Note: please replace TBD1, TBD2, with the assigned values, throughout the document.

2.4.1.1. largeFlowObservationInterval

Description:

The minimum time interval to observe a flow for performing further processing of the flow.

Abstract Data Type: unsigned64

ElementId: TBD1

Units: milliseconds

Status: Current

2.4.1.2. largeFlowBandwidthThreshold

Description:

The minimum bandwidth of the flow during the observation interval (largeFlowObservationInterval) for declaring the flow a large flow. Unit is in Mbps.

Abstract Data Type: unsigned64

ElementId: TBD2

Units: Mbps

Status: Current

3. Current sampling techniques for security threat detection

Packet sampling techniques e.g. PSAMP -- [RFC 5474], [RFC 5475], [RFC 5476], [RFC 5477], in switches and routers provide an effective mechanism for approximate detection of various types of flows -- long-lived large flows and other flows (which include long-lived small flows, short-lived small/large flows) with minimal packet replication bandwidth overhead. The packet sampling techniques sample all flows equally.

A large percentage of the packet samples comprise of long-lived large (aka large) flows and a small percentage of the packet samples comprise of other (aka small) flows. The large flows aka top-talkers consume a large percentage of the bandwidth and small percentage of the flow space.

The small flows, which are the typical cause of security threats like Denial of Service (DOS) attacks, scanning attacks etc., consume a small percentage of the bandwidth and a large percentage of the flow space.

4. Application of flow-state dependent packet selection techniques for security threat detection

Using the flow-state dependent packet selection techniques described in Section 2, the large flows or top-talkers can be detected in real-time with a high degree of accuracy. Only the small flows need to be sampled -- this makes security threat detection more effective with minimal sampling overhead.

The steps in security threat detection are described below

1) Large Flow Identification:

For identifying large flows, use the flow-state dependent packet selection techniques described in Section 2. This helps in identifying the large flows aka top-talkers in real-time with a high degree of accuracy.

2) Large Flow Classification:

The identified large flows can be broadly classified into 2 categories as detailed below.

- a. Well behaved (steady rate) large flows, e.g. video streams
- b. Bursty (fluctuating rate) large flows e.g. Peer-to-Peer traffic

The large flows can be sampled at a low rate for further analysis or need not be sampled. If desired, the large flows could be exported to a central entity, e.g. Netflow Collector, using IPFIX protocol [RFC 5101] for further analysis.

3) Small Flow Processing:

The small flows (excluding the large flows) can be sampled at a normal rate. The small flows can be examined for determining security threats like DOS attacks (for e.g. SYN floods), Scanning attacks etc. [FDDOS, PDSN, and ALDS]

Thus, we can see that, security threat detection is possible with minimal sampling overhead.

4.1. Analysis of various flow-state dependent packet selection techniques

The multistage filter technique suggested in [ESVA] for automatic identification works well for standard applications generating large flows, for e.g. video content like movies and catch-up episodes, backup transactions etc. with a detection time of approximately 30-60

seconds. These detection times ensure that short-lived large flows, for e.g. HD video clips, are not unnecessarily recognized.

If faster large flow identification times are desired (much shorter than 30s), the multistage filter technique suggested in [ESVA] may pose the following problem that the effective filtered flow size is phase-dependent: that is, relatively smaller constant-rate flows, for e.g. HD video clips, beginning early within a counting Bloom filter reset interval would be unnecessarily detected with the same probability as relatively larger flows beginning toward the interval. [VRM] suggests techniques for addressing the above problem using rotating conservative counting Bloom filters with periodic decay.

4.2. Simulation

Simulation results for these flow-state dependent packet selection techniques are presented in Appendix A. The goal of the simulation is to demonstrate the effectiveness of these techniques for security threat detection in a multi-tenant video streaming data center.

5. Security Considerations

This document does not directly impact the security of the Internet infrastructure or its applications. In fact, it proposes techniques which could help in identifying a DOS attack pattern.

6. Operational Considerations

For effectively using the flow-state dependent packet selection techniques, the operator should adjust the programmable parameters `largeFlowObservationInterval` and `largeFlowBandwidthThreshold` in switches/routers based on the applications which are being deployed.

7. Acknowledgements

The authors would like to thank Juergen Quittek, Brian Carpenter, Michael Fargano, Michael Bugenhagen, Jianrong Wong, Brian Trammell and Paul Aitken for all the support and valuable input.

8. References

8.1. Normative References

8.2. Informative References

[RFC 5474] N. Duffield et al., "A Framework for Packet Selection and Reporting", March 2009.

[RFC 5475] T. Zseby et al., "Sampling and Filtering Techniques for IP Packet Selection", March 2009.

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

[RFC 5477] T. Dietz et al., "Information Model for Packet Sampling Exports", March 2009.

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

[RFC 6728] G. Muenz et al., "Configuration Data Model for the IP Flow Information Export (IPFIX) and Packet Sampling (PSAMP) Protocols"

[VRM] G. Bianchi et al., "Measurement Data Reduction through Variation Rate Metering", INFOCOM 2010

[PDSN] Ignasi Paredes-Oliva et al., "Portscan Detection with Sampled NetFlow", TMA 2009

[ALDS] Z. Morley Mao et al., "Analyzing Large DDoS Attacks Using Multiple Data Sources", SIGCOMM 2006

[FDDOS] David Holmes, "The DDoS Threat Spectrum", F5 White paper 2012

[ESVA] C. Estan and G. Varghese, "New Directions in Traffic Measurement and Accounting", ACM SIGCOMM Internet Measurement Workshop 2001, San Francisco (CA) Nov. 2001.

Appendix A: Simulation of Flow aware packet sampling

Goal:

Demonstrate the effectiveness of flow aware packet sampling in a practical use case, for e.g. multi-tenant video streaming in a data center.

Test Topology:

Multiple virtual servers (server hosted on a virtual machine) connected to a virtual switch (vSwitch) which in turn connects to the data center network using a 10Gbps ethernet interface.

2 virtual servers are active.

First virtual server

- . Traffic types
 - o HD MPEG-4 video streams (bit rate 10Mbps) - 100 - 1Gbps
 - o SD MPEG-2 video streams (bit rate 4Mbps) - 300 - 1.2Gbps
 - o Other traffic - 500Mbps (Video clips, DOS attacks (for e.g. SYN floods), Scanning attacks etc.)
- . Aggregate traffic - 2.7Gbps

Second virtual server

- . Traffic types
 - o HD MPEG-4 video streams (bit rate 10Mbps) - 50 - .5Gbps
 - o SD MPEG-2 video streams (bit rate 4Mbps) - 500 - 2.0Gbps
 - o Backup transaction - 100Mbps
 - o Other traffic - 500Mbps (Video clips, DOS attacks (for e.g. SYN floods), Scanning attacks etc.)
- . Aggregate traffic - 3.1Gbps

Total traffic on 2 servers - 5.8Gbps

Existing techniques:

Normal sampling rate - 1:1000

Total sampled traffic = $5.8\text{Gbps}/1000 = 5.8\text{Mbps}$

Flow aware sampling technique:

Large flow recognition parameters

- . Observation interval for large flow - 60 seconds
- . Minimum bandwidth threshold over the observation interval - 2Mbps

Aggregate bit rate of large flows = 4.8Gbps

Aggregate bit rate of small flows = 1Gbps

Low sampling rate of large flows - 1:10000

Normal sampling rate of small flows - 1:1000

Total sampled traffic = $4.8\text{Gbps}/10000 + 1\text{Gbps}/1000 = 1.48\text{Mbps}$

Percentage improvement in sampling (most of the samples are only small flows) = $(5.8 - 1.48)/5.8 \approx 78\%$

The small flows can be examined in a central entity like Netflow Collector for determining security threats like DOS attacks, Scanning attacks etc. Thus, we can see that, security threat detection is possible with minimal sampling overhead.

Authors' Addresses

Ram Krishnan
Brocade Communications
San Jose, 95134, USA

Phone: +001-408-406-7890
Email: ramk@brocade.com

Ning So
Tata Communications
Plano, TX 75082, USA

Phone: +001-972-955-0914
Email: ning.so@tatacommunications.com

IPFIX Working Group
Internet-Draft
Intended status: Informational
Expires: May 9, 2013

B. Trammell
ETH Zurich
November 5, 2012

Textual Representation of IPFIX Abstract Data Types
draft-trammell-ipfix-text-adt-00.txt

Abstract

This document defines UTF-8 representations for IPFIX abstract data types, to support interoperable usage of the IPFIX Information Elements with protocols based on textual encodings.

Status of this Memo

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

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

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

This Internet-Draft will expire on May 9, 2013.

Copyright Notice

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

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

Table of Contents

1. Introduction	3
2. Terminology	3
3. Identifying Information Elements	3
4. Data Type Encodings	4
4.1. octetArray	4
4.2. unsigned*	4
4.3. signed*	5
4.4. float*	6
4.5. boolean	6
4.6. macAddress	6
4.7. string	6
4.8. dateTime*	6
4.9. ipv4Address	7
4.10. ipv6Address	7
4.11. basicList, subTemplateList, and subTemplateMultiList	7
5. Security Considerations	7
6. IANA Considerations	7
7. References	7
7.1. Normative References	7
7.2. Informative References	8
Appendix A. Example	8
Author's Address	10

1. Introduction

The IPFIX Information Model, as defined by the IANA IPFIX Information Element Registry, provides a rich set of Information Elements for description of information about network entities and network traffic data, and abstract data types for these Information Elements. The IPFIX Protocol Specification [I-D.ietf-ipfix-protocol-rfc5101bis], in turn, defines a big-endian binary encoding for these abstract data types suitable for use with the IPFIX Protocol.

However, present and future operations and management protocols and applications may use textual encodings, and generic framing and structure as in JSON or XML. A definition of canonical textual encodings for the IPFIX abstract data types would allow this set of Information Elements to be used for such applications, and for these applications to interoperate with IPFIX applications at the Information Element definition level.

Note that templating or other mechanisms for data description for such applications and protocols are application specific, and therefore out of scope for this document: only Information Element identification and data value representation are defined here.

2. Terminology

Capitalized terms defined in the IPFIX Protocol Specification [I-D.ietf-ipfix-protocol-rfc5101bis] and the IPFIX Information Model [I-D.ietf-ipfix-information-model-rfc5102bis] are used in this document as defined in those documents. In addition, this document defines the following terminology for its own use:

Enclosing Context

Textual representation of IPFIX data values is applied to use the IPFIX Information Model within some existing textual format (e.g. XML, JSON). This outer format is referred to as the Enclosing Context within this document. Enclosing Contexts define escaping and quoting rules for represented data values.

3. Identifying Information Elements

The IPFIX Information Element Registry [iana-ipfix-assignments] defines a set of Information Elements and numbered by Information Element Identifiers, and named for human-readability. These Information Element Identifiers are meant for use with the IPFIX protocol, and have little meaning when applying the IPFIX Information Element Registry to textual representations.

Instead, applications using textual representations of Information Elements SHOULD use Information Element names to identify them; see Appendix A for examples illustrating this principle.

4. Data Type Encodings

[FIXME frontmatter]

This section uses ABNF [RFC5234], including the Core Rules in Appendix B, to describe the format of textual representations of IPFIX abstract data types.

4.1. octetArray

[FIXME: native hex strings for comparative human readability.]

4.2. unsigned*

First, in the special case that the unsigned Information Element has identifier semantics, and refers to a set of codepoints, either in an external registry, a sub-registry, or directly in the description of the Information Element, then the name or short description for that codepoint MAY be used to improve readability.

If the Enclosing Context defines a representation for unsigned integers, that representation SHOULD be used.

Otherwise, the values of Information Elements of an unsigned integer type may be represented either as unprefixd base-10 (decimal) strings, or as base-16 (hexadecimal) strings prefixed by '0x'; in ABNF:

```
unsigned = 1*DIGIT / '0x' 1*HEXDIG
```

Leading zeroes are allowed in either encoding, and do not signify base-8 (octal) encoding.

The encoded value must be in range for the corresponding abstract data type or Information Element. Out of range values should be interpreted as clipped to the implicit range for the Information Element as defined by the abstract data type, or to the explicit range of the Information Element if defined. Minimum and maximum values for abstract data types are shown in Table 1 below.

type	minimum	maximum
unsigned8	0	255
unsigned16	0	65536
unsigned32	0	4294967295
unsigned64	0	18446744073709551615

Table 1: Ranges for unsigned abstract data types

4.3. signed*

If the Enclosing Context defines a representation for signed integers, that representation should be used.

Otherwise, the values of Information Elements of signed integer types should be represented as optionally-prefixed base-10 (decimal) strings; if the sign is omitted, it is assumed to be positive. In ABNF:

```
sign = "+" / "-"
```

```
signed = [sign] 1*DIGIT
```

Leading zeroes are allowed, and do not signify base-8 (octal) encoding.

The encoded value must be in range for the corresponding abstract data type or Information Element. Out of range values should be interpreted as clipped to the implicit range for the Information Element as defined by the abstract data type, or to the explicit range of the Information Element if defined. Minimum and maximum values for abstract data types are shown in Table 2 below.

type	minimum	maximum
signed8	-128	+127
signed16	-32768	+32767
signed32	-2147483648	+2147483647
signed64	-9223372036854775808	+9223372036854775807

Table 2: Ranges for signed abstract data types

4.4. float*

If the Enclosing Context defines a representation for floating point numbers, that representation should be used.

[FIXME: there appears to be no defined (non-interchange) format for floating point numbers, but we probably want to define something reasonably human-readable without getting too into locale issues.]

exponent = 'e' 1*3DIGIT

right-decimal = '.' 0*DIGIT

float = [sign] 1*DIGIT [right-decimal] [exponent]

4.5. boolean

[FIXME: frontmatter. note that booleans may also be naturally represented by the presence or absence of a value in the structure of the document in the Enclosing Context.]

boolean-yes = "l" / "y" / "Y" / "t" / "T"

boolean-no = "0" / "n" / "N" / "f" / "F"

boolean = boolean-yes / boolean-no

4.6. macAddress

[FIXME: frontmatter]

macaddress = 2*HEXDIG 5*(":" 2*HEXDIG)

4.7. string

As Information Elements of the string type are simply UTF-8 encoded strings, they are represented directly, subject to the escaping and encoding rules of the Enclosing Context. If the Enclosing Context cannot natively represent UTF-8 characters, the escaping facility provided by the Enclosing Context must be used for non-representable characters. Additionally, strings containing characters reserved in the Enclosing Context (e.g. markup characters, quotes) must be escaped or quoted according to the rules of the Enclosing Context.

4.8. dateTime*

[FIXME: [RFC3339]]

[FIXME: explain precision rules]

4.9. ipv4Address

[FIXME: frontmatter. dotted-quad.]

ipv4address = 1*3DIGIT 3*("." 1*3DIGIT)

4.10. ipv6Address

[FIXME: section 2.2 of [RFC4291], recommend section 4 of [RFC5952]]

4.11. basicList, subTemplateList, and subTemplateMultiList

These abstract data types, defined for IPFIX Structured Data [RFC6313], do not represent actual data types; they are instead designed to provide a mechanism by which complex structure below the template level. It is assumed that protocols using textual Information Element representation will provide their own structure. Therefore, Information Elements of these Data Types should not be used in textual representations.

5. Security Considerations

[FIXME: content would be nice]

6. IANA Considerations

This document has no considerations for IANA.

7. References

7.1. Normative References

- [I-D.ietf-ipfix-protocol-rfc5101bis]
Claise, B. and B. Trammell, "Specification of the IP Flow Information eXport (IPFIX) Protocol for the Exchange of Flow Information", draft-ietf-ipfix-protocol-rfc5101bis-02 (work in progress), June 2012.
- [RFC3339] Klyne, G., Ed. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, July 2002.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.

- [RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.
- [RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, August 2010.
- [iana-ipfix-assignments]
Internet Assigned Numbers Authority, "IP Flow Information Export Information Elements
(<http://www.iana.org/assignments/ipfix/ipfix.xml>)",
November 2012.

7.2. Informative References

- [I-D.ietf-ipfix-information-model-rfc5102bis]
Claise, B. and B. Trammell, "Information Model for IP Flow Information eXport (IPFIX)",
draft-ietf-ipfix-information-model-rfc5102bis-06 (work in progress), October 2012.
- [I-D.ietf-ipfix-ie-doctors]
Trammell, B. and B. Claise, "Guidelines for Authors and Reviewers of IPFIX Information Elements",
draft-ietf-ipfix-ie-doctors-07 (work in progress),
October 2012.
- [RFC6313] Claise, B., Dhandapani, G., Aitken, P., and S. Yates,
"Export of Structured Data in IP Flow Information Export (IPFIX)", RFC 6313, July 2011.

Appendix A. Example

In this section, we examine an IPFIX Template and a Data Record defined by that Template, and show how that Data Record would be represented in JSON according to the specification in this document. Note that this is specifically NOT a recommendation for a particular representation, merely an illustration of the encodings in this document.

[FIXME improve frontmatter] Figure 1 shows a Template in IESpec format as defined in section 9.1 of [I-D.ietf-ipfix-ie-doctors]. A Message containing this Template and a Data Record is shown in Figure 2, and a corresponding JSON Object using the text format defined in this document is shown in Figure 3.

```

flowStartMilliseconds(152)<dateTimeMilliseconds>[8]
flowEndMilliseconds(153)<dateTimeMilliseconds>[8]
octetDeltaCount(1)<unsigned64>[4]
packetDeltaCount(2)<unsigned64>[4]
sourceIPv6Address(27)<ipv4Address>[4]{key}
destinationIPv6Address(28)<ipv4Address>[4]{key}
sourceTransportPort(7)<unsigned16>[2]{key}
destinationTransportPort(11)<unsigned16>[2]{key}
protocolIdentifier(4)<unsigned8>[1]{key}
tcpControlBits(6)<unsigned8>[1]
flowEndReason(136)<unsigned8>[1]

```

Figure 1: Sample flow template (IPFIX)

1										2										3										4										5										6																				
0	2	4	6	8	0	2	4	6	8	0	2	4	6	8	0	2	4	6	8	0	2	4	6	8	0	2	4	6	8	0	2	4	6	8	0	2																																		
0x000a										length 135										export time 1352140263																				msg																														
sequence 0																				domain 1																				hdr																														
SetID 2										length 52										tid 256										fields 11										tmpl																														
IE 152										length 8										IE 153										length 8										set																														
IE 1										length 4										IE 2										length 4																																								
IE 27										length 16										IE 28										length 16																																								
IE 7										length 2										IE 11										length 2																																								
IE 4										length 1										IE 6										length 1																																								
IE 136										length 1										SetID 256										length 83										data																														
start time																														1352140261135										set																														
end time																														1352140262880																																								
octets										195383										packets										88																																								
sip6																																																																						
																				2001:0db8:000c:1337:0000:0000:0000:0002																																																		
dip6																																																																						
																				2001:0db8:000c:1337:0000:0000:0000:0003																																																		
sp										80										dp										32991										prt 6										tcp 19										fe 3										

Figure 2: IPFIX message containing sample flow


```
{
  "flowStartMilliseconds": "2012-11-05 18:31:01.135",
  "flowEndMilliseconds": "2012-11-05 18:31:02.880",
  "octetDeltaCount": 195383,
  "packetDeltaCount": 88,
  "sourceIPv6Address": "2001:db8:c:1337::2",
  "destinationIPv6Address": "2001:db8:c:1337::3",
  "sourceTransportPort": 80,
  "destinationTransportPort": 32991,
  "protocolIdentifier": "tcp",
  "tcpControlBits": 19,
  "flowEndReason": 3
}
```

Figure 3: JSON object containing sample flow

Author's Address

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

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

