

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:

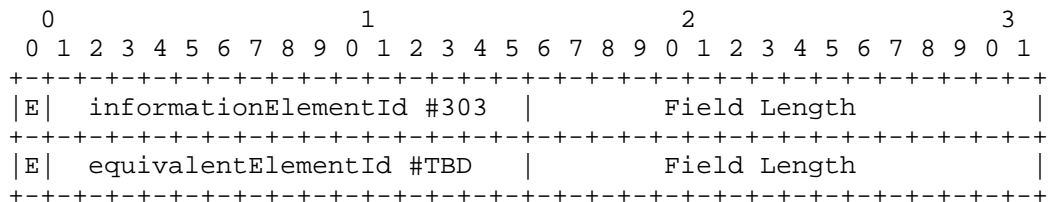


Figure 1: Template for Equivalence Message

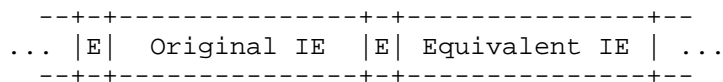


Figure 2: Equivalence Message Data Record

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 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


```

+-----+
|      Field Length = 65536      |
+-----+

```

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 Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 13, 2012

B. Claise
P. Aitken
S. B S
Cisco Systems, Inc.
J. Schoenwaelder
Jacobs University Bremen
March 12, 2012

Exporting MIB Variables using the IPFIX Protocol
draft-johnson-ipfix-mib-variable-export-04

Abstract

This document specifies a way to complement IPFIX Flow Records with Management Base (MIB) objects, avoiding the need to define new IPFIX Information Elements for existing Management Information Base objects that are already fully specified.

This method requires an extension to the current IPFIX protocol. New Template Set and Options Template Sets are specified to allow the export of Simple Network Management Protocol (SNMP) MIB Objects along with IPFIX Information Elements.

Status of this Memo

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

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

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

This Internet-Draft will expire on September 13, 2012.

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. Open Issues / To do list	5
2. Introduction	6
3. Motivation and Architectural Model	7
4. Terminology	9
5. MIB OID Extended Template Formats	10
5.1. MIB OID Extended Template Record Format	10
5.2. MIB OID Extended Options Template Record Format	11
5.3. MIB OID Extended Field Specifier Format	12
5.3.1. Standard Field Specifier Format	13
5.3.2. Extended Field Specifier Format for a non-indexed MIB Object	14
5.3.3. Extended Field Specifier Format for an Indexed MIB Object, with an MIB OID as Index	15
5.3.4. Extended Field Specifier Format for an Indexed MIB Object, with an IPFIX Information Element as Index	18
5.3.5. Extended Field Specifier Format for an Indexed MIB Object, with one IPFIX Information Element for the OID segment identifying the instance	20
5.4. Indices Considerations	23
5.5. Identifying the SNMP Context	24
5.6. Template Management	24
6. Example Use Cases	24
6.1. Without Using the Specifications in this Document	25
6.2. Non-indexed MIB Object: Established TCP Connections	25
6.3. Enterprise Specific MIB Object: Detailing CPU Load History	28
6.4. Indexed MIB Object with an OID: Output Interface Queue Size in PSAMP Packet Report	30
6.5. Indexed MIB Object with Two OIDs: The ipIfStatsInForwDatagrams	34
6.6. Indexed MIB Object with an IPFIX Information Element: Output Interface Queue Size in PSAMP Packet Report	36
6.7. Indexed MIB Objects with a mix of MIB OID and IPFIX Information Element	40
6.8. Indexed MIB Object with MIBInstanceIdentifier Information Element: ipIfStatsOutOctets	40
6.9. Using MIB Objects as IPFIX Options Scope fields	42
6.9.1. Using non-Indexed MIB Objects as Option Scope fields	42
6.9.2. Using Indexed MIB Objects as Option Scope fields	44
6.10. Using MIB Objects with IPFIX Structured Data	46
7. Configuration Considerations	47
8. The Collecting Process's Side	47
9. Applicability	47
10. Security Considerations	48
11. IANA Considerations	48

11.1. New Set IDs	48
11.2. New Data Types	48
11.3. New Information Elements	48
12. Acknowledgements	49
13. References	49
13.1. Normative References	49
13.2. Informative References	50
Authors' Addresses	51

1. Open Issues / To do list

- o Skipping the length. Is a new Set ID the right solution?
- o "timestamps, exporters, and other animals" -> see the mailing list.
- o Question: index is an IPFIX IE that didn't appear the flow record? Do we preclude this case?
- o The value of the MIB OID acting as an index may not be of fixed length and may have no default length, for example the OID can be of type string or type MIB OID.
- o "we can use the IE as an index if there is one and only one similar with that length in the Template Records". To be discussed.
- o use case: no index count and no index OID in the SNMP agent -> add this with the solution discussed with the DCM2.0 team.
- o This also allows reduced size encoding for the indices.
- o some TODO in the XML version:
 - * write section: "Indexed MIB Objects with a mix of MIB OID and IPFIX Information Element"
 - * insert example: "Using MIB Objects with IPFIX Structured Data"
- o Describe how to choose between multiple instances of the required index field (eg, when the index is the egress interface for multicast). eg, rather than specifying the index IE by ID, we could specify it by number: the n'th field in the record.
- o IPFIX Structured Data: how should it work? Add example to "sectionStructuredData".
- o How does the example in 5.5 work (ifOutQLen indexed by: ifIndex) since ifIndex is not present in the record?
- o How does the example in 5.8.2 work, since the ifName is indexed by ifIndex which comes after - so the value is not already known.
- o Improve the examples: Add an example with the mix of IPFIX IE and OID in sectionUseIndexedwithaMixofOIDAndIPFIXIE.

- o RFC 5610: explain what needs to be updated.
- o ID to name mappings? -> use this for an example in section 5.
- o What does this mean? : "(Consider the counter synchronization issue, non-key info should be static)".
- o Tidy up the XML.
- o (JS) Do we need to add something about the contextEngineID and contextName? Optionally associate context with template via options Could be done with common properties or in a flow record However, do we limit all MIB variables in a Template Record to a single context? 3 cases:
 1. if a simple SNMP agent, no contextEngineID and contextName, because it's the default
 2. the context information is valid for the entire flow record
 3. the context information is specific for each IE within the entire flow record

question regarding 3.: only one context for an entire flow or can a flow record export MIB OID from different context? (JS): ask the IPFIX mailing list. (BC): ask internally in Cisco Action: complete the "Identifying the SNMP Context" section
- o (JS) Inacio's figure: send email to the mailing list.

2. Introduction

There is growing interest in using IPFIX as a push mechanism for exporting management information. Using a push protocol such as IPFIX instead of a polling protocol like SNMP is especially interesting in situations, where large chunks of repetitive data need to be exported periodically.

While initially targeted at different problems, there is a large parallel between the information transported via IPFIX and SNMP. Furthermore, certain Management Information Base (MIB) objects are highly relevant to flows as they are understood today. For example, in the IPFIX information model [RFC5102], Information Elements coming from the SNMP world have already been specified, e.g., ingressInterface and egressInterface both refer to the ifIndex defined in [RFC2863].

Rather than mapping existing MIB objects to IPFIX Information Elements on a case by case basis, it would be advantageous to enable the export of any existing or future MIB objects as part of an IPFIX Flow Record. This way, the duplication of data models [RFC3444], both as SMI MIB objects and IPFIX Information Elements, out of the same information model [RFC3444] would be avoided.

In this document, new Template Sets for Flow Records and Options Records are specified to allow Templates to contain any combination of fields defined by traditional IPFIX Information Element(s) and/or MIB Object Identifier(s). The MIB Object Identifiers can reference either non-indexed or indexed MIB object(s). Note that the enterprise-specific MIB Object Identifiers are also supported.

When an indexed MIB object is exported, a method to identify how that MIB object is indexed is specified so that the full meaning of the information being exported can be conveyed. The specifications encompasses the different index types for the MIB Objects Identifier: indexed by one or multiple MIB variable(s), indexed by one or multiple IPFIX Information Element(s), indexed by a mix of MIB variable(s) and IPFIX Information Element(s). A set of example use cases is used to illustrate how these specifications can be used.

Some Exporters may not have the knowledge to convey the full information on how the MIB objects being exported are indexed. They may not know the index count and/or the OID's of the objects that are used to index a MIB object. In such cases the Exporter can send the the values of the index OID's identifying the instance of the object being exported as one string that conveys the instance identifier part of an object being exported. The Collecting Process may know how a MIB object is indexed by some other means, for example, it could compile this information from the MIB Module that defines exported MIB object or the Collecting Process could be hardcoded with this information for a pre-defined set of MIB objects that it is interested in. An example use case is used to illustrate this mechanism.

3. Motivation and Architectural Model

Most Flow Records contain the ingressInterface and/or the egressInterface Information Element. These Information Elements carry an ifIndex value, a MIB object defined in [RFC2863]. In order to retrieve additional information about the identified interface, a Collector could simply poll relevant objects from the device running the Exporter via SNMP, however, that approach has several problems:

- o It requires implementing a mediation function between two data models, i.e., MIB objects and IPFIX Information Elements.
- o Confirming the validity of simple mappings (e.g., ifIndex to ifName) requires to either check on a regular basis that the Exporter's network management system did not reload, or to impose ifIndex persistence across an Exporter's reload.
- o Synchronization problems occur since counters carried in Flow Records and counters carried in SNMP messages are retrieved from the Exporter at different points in time and thus can't be correlated. In the best case, assuming very tight integration of an IPFIX Collector with an SNMP polling engine, SNMP data is retrieved shortly after Data Records have been received, which implies the sum of the active or inactive timeouts (if not null) plus the time to export the Flow Record to the Collector. If, however, the SNMP data is retrieved by a generic Network Management Station (NMS) polling interface statistics, then the time lag between IPFIX counters and SNMP counters can be significant.

The intended scope of this work is the addition of MIB variable(s) to IPFIX Information Elements in Flow Records, in order to complement the Flow Records with useful and already standardized information. More specifically, the case of an existing Template Record, which needed to be augmented with some MIB variables whose index was already present in the Template Record as an IPFIX Information Element: typically, a 7-tuple Flow Record containing the ingressInterface Information Element, augmented by interface counters [RFC2863], which are indexed by the respective ingressInterface values in the Flow Records.

The intended goal of this work is not a replacement of SNMP notifications, even if the specifications in this document could potentially allow this. Since IPFIX is a push mechanism, initiated from the Exporter with no acknowledgment method, this specification does not provide the ability to execute configuration changes.

The Distributed Management Expression MIB [RFC2982], which is a mechanism to create new MIB variables based on the content of existing ones, could also be advantageous in this context of this specification. Indeed, newly created MIB object (for example, the link utilization MIB variable), created with the Distributed Management Expression MIB [RFC2982] could nicely complement Flow Records.

Another advantage of exporting MIB objects via IPFIX is that IPFIX would benefit from an extended series of types to be exported. The

simple and application-wide data types specified in SMIV2 [RFC2578], along with a new textual conventions, can be exported within IPFIX and then decoded in the Collector.

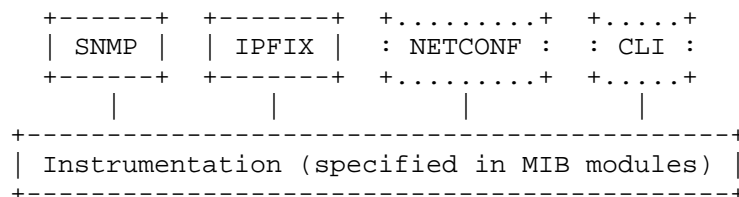


Figure 1: Architectural Model

The overall architectural model is depicted in Figure 1. The IPFIX Exporter accesses the device's instrumentation, which follows the specifications contained in MIB modules. Other management interfaces such as NETCONF or the device's Command Line Interface (CLI) may provide access to the same instrumentation.

4. Terminology

IPFIX-specific terminology (Information Element, Template, Template Record, Options Template Record, Template Set, Collector, Exporter, Flow Record, etc.) used in this document is defined in Section 2 of [RFC5101]. As in [RFC5101], these IPFIX-specific terms have the first letter of a word capitalized.

This document prefers the more generic term "Data Record" as opposed to "Flow Record" as this specification allows the export of MIB objects.

MIB Object Identifier (MIB OID)

An ASCII character sequences of decimal non-negative sub-identifier values. Each sub-identifier value MUST NOT exceed $2^{32}-1$ (4294967295) and MUST NOT have leading zeros. Sub-identifiers are separated by single dots and without any intermediate whitespace.

MIB Object Identifier Information Element

An IPFIX Information Element ("MIBObjectIdentifierMarker") that denotes that a MIB Object Identifier is exported in the (Options) Template Record.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",

"SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

5. MIB OID Extended Template Formats

Extended Template Record Formats are required to export data defined by MIB Object Identifiers. New Template Sets are required for these extended Template Record Formats.

5.1. MIB OID Extended Template Record Format

The format of the MIB Object Identifier Extended Template Record is shown in Figure 2. It consists of a Template Record Header and one or more Field Specifiers.

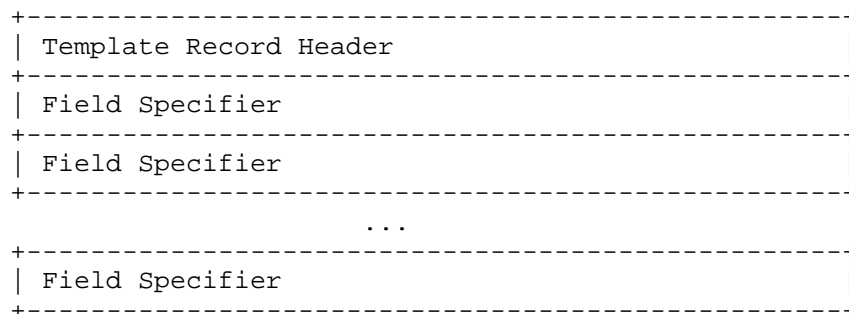


Figure 2: MIB Object Identifier Extended Template Record Format

A MIB Object Identifier Extended Template Record MUST contain at least one MIB Object Identifier Extended Field Specifier. It MAY also contain any combination of IANA-assigned and/or enterprise-specific Information Element identifiers as specified in [RFC5101].

The format of the Template Record Header is shown in Figure 3.

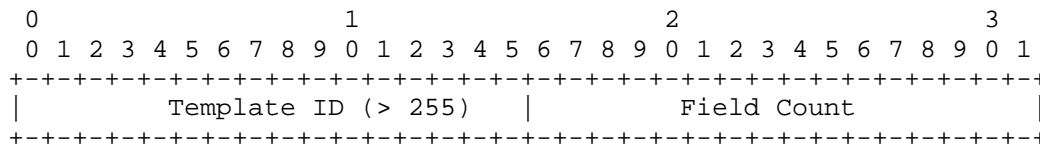


Figure 3: Template Record Header Format

Where:

Template ID

Template ID of this Template Record. This value is greater than 255.

Field Count

Number of all fields in this Template Record.

At this level of detail the layout of the Template Record Format, as specified in [RFC5101], and the MIB Object Identifier Extended Template Record Format are identical. It is only the structure of the Field Specifiers that is different (see Section 5.3).

5.2. MIB OID Extended Options Template Record Format

The format of the MIB Object Identifier Extended Options Template Record is shown in Figure 4. It consists of an Options Template Record Header and one or more Field Specifiers.

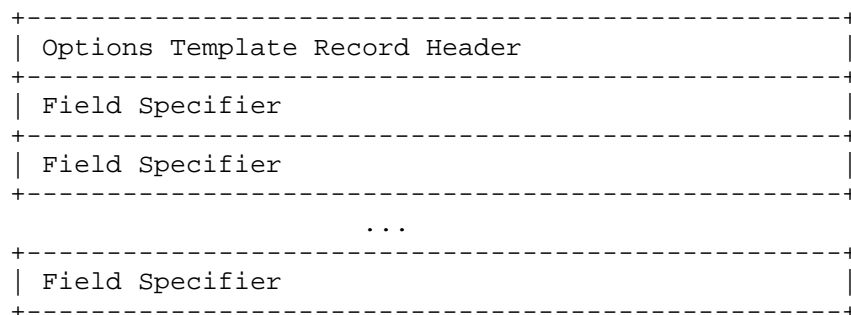


Figure 4: MIB Object Identifier Options Extended Template Record Format

A MIB Object Identifier Extended Options Template Record MUST contain at least one MIB Object Identifier Extended Field Specifier, which MAY be a scope field. It MAY also contain any combination of IANA-assigned and/or enterprise-specific Information Element identifiers.

The format of the Options Template Record Header is shown in Figure 5.

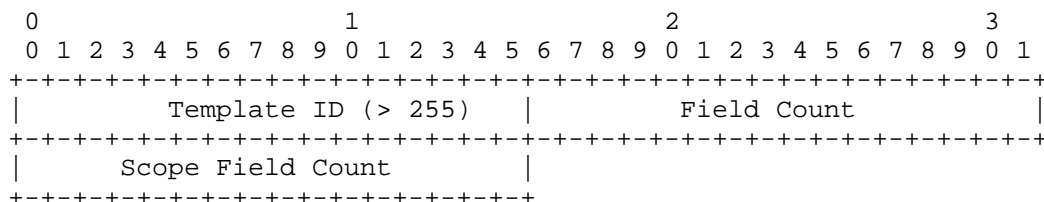


Figure 5: Options Template Record Header Format

Where:

Template ID

Template ID of this Template Record. This value is greater than 255.

Field Count

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

Scope Field Count

Number of scope fields in this Options Template Record. The Scope Fields are normal Fields except that they are interpreted as Scope at the Collector. The Scope Field Count MUST NOT be zero for an Options Template Record.

As with the Template Record Format, the only difference between the standard Options Template Record Format as defined in [RFC5101] and the MIB Object Identifier Extended Template Options Record Format is the structure of the Field Specifiers (see Section 5.3).

Both indexed and non-indexed MIB Objects may be used as scope fields in an IPFIX Options Template Record. Each scope MIB object is included in the IPFIX Scope Field Count. When indexed MIB Objects are used, the index information is not included in the Scope Field Count since the size of the index information is already specified in the MIB Object's "index count" field (see Section 5.3.3). Examples are given in Section 6.9.

5.3. MIB OID Extended Field Specifier Format

This section specifies how the Field Specifier format in [RFC5101] is extended to allow fields to be defined using a specified MIB Object. First for a MIB Object Identifier that is a non-indexed MIB object, then for an indexed MIB object.

The Field Specifier formats are shown in Figure 6 to Figure 9 below.

5.3.1. Standard Field Specifier Format

The Field Specifier format in Figure 6, along with the associated definitions, has been copied from [RFC5101], for an easier comparison with the MIB Object Identifier Extended Field Specifier Format in Figure 7 through Figure 9.

When exporting an IANA-assigned and/or enterprise-specific IPFIX Information Element identifier, the Field Specifier Format is the same as shown below.

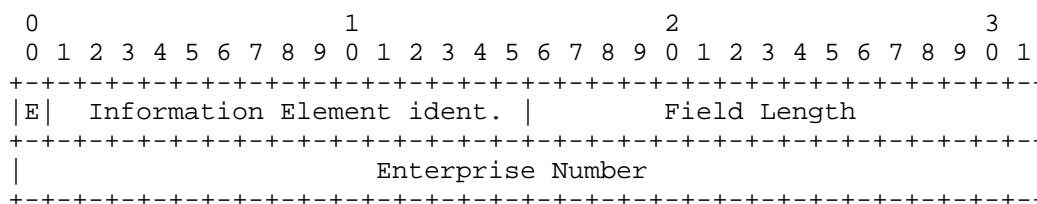


Figure 6: Standard Field Specifier format

Where:

E

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

Information Element identifier

A numeric value that represents the type of the Information Element. Refer to [RFC5102].

Field Length

The length of the corresponding encoded Information Element, in octets. Refer to [RFC5102]. The field length may be smaller than the definition in [RFC5102] if reduced size encoding is used. The value 65535 is reserved for variable length Information Element.

Enterprise Number

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

5.3.2. Extended Field Specifier Format for a non-indexed MIB Object

When a MIB object is to be exported, a special Information Element value is used to show that the extended Field Specifier is being used, as shown in Figure 7:

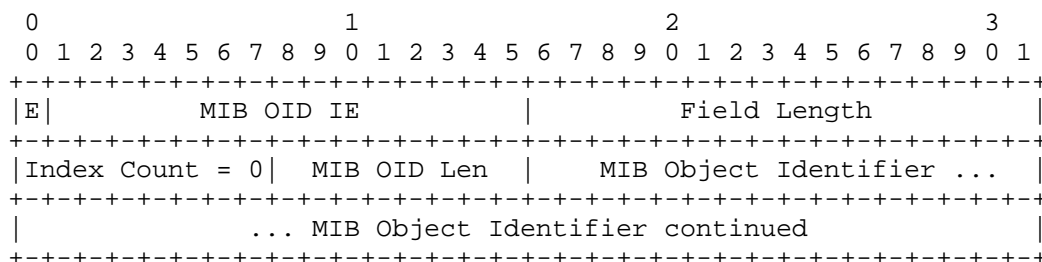


Figure 7: MIB Object Identifier Extended Field Specifier Format for a non-indexed MIB Object with an OID length < 255

Where:

E

Enterprise bit. This is the first bit of the Field Specifier. The value is always set to 0 for the MIB Object Identifier Extended Field Specifier Format, even if the MIB Object Identifier is enterprise-specific, because the MIB OID IE is an IANA standard field and is not enterprise-specific.

MIB OID IE

Special IPFIX Information Element, MIBObjectIdentifierMarker, that denotes that a MIB object is exported in the (Options) Template Record. When the MIB Object Identifier Information Element (MIB OID IE) is used, the MIB Object Identifier must be specified in the MIB Object Identifier Extended Field Specifier for the Collecting Process to be able to decode the Records.

Field Length

The definition is as [RFC5101]. Note that the Field Length can be expressed using reduced size encoding per [RFC5101].

Index Count

The number of indices for a MIB object. Set to zero for a non-indexed MIB object.

MIB Object Identifier Length

The length of the textual representation of the MIB Object Identifier that follows. This is encoded in the same manner as the variable length encoding in [RFC5101]. If the length of the MIB Object Identifier is greater than or equal to 255 octets, the length is encoded into 3 octets before the MIB Object Name, where the first octet is 255 and the length is carried in the second and third octets as shown in Figure 8. If the MIB Object Identifier is longer than 254 characters then the length MUST be extended.

MIB Object Identifier

The textual representation of a MIB object identifier as defined in Section 4.

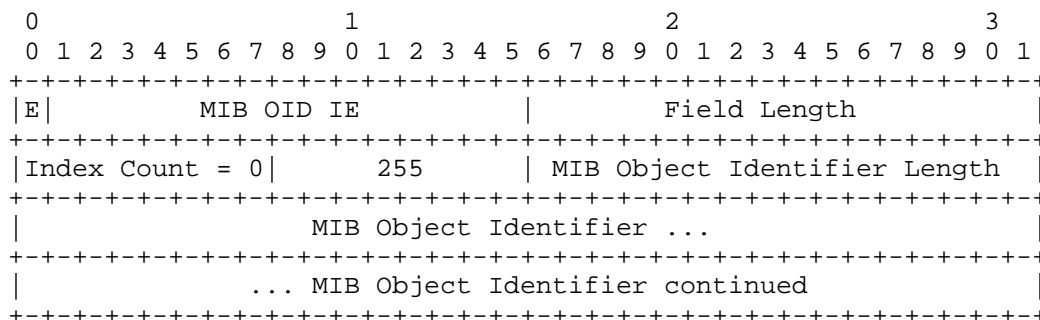


Figure 8: MIB Object Identifier Extended Field Specifier Format for a non-indexed MIB Object with an OID length >= 255

5.3.3. Extended Field Specifier Format for an Indexed MIB Object, with an MIB OID as Index

The mechanism for "Extended Field Specifier Format for non-indexed MIB Object" in Section 5.3.2 can be used for exporting any MIB objects, including indexed MIB objects. However, per the nature of indexing in MIB module, every indexed object is specified by a new MIB Object Identifier, which in turn implies that a new Template Record must be used for every indexed object. For example, the ifInOctets for the interface represented by the interface ifIndex 1

is ifInOctets.1, the ifInOctets for the interface represented by the interface ifIndex 2 is ifInOctets.2, ... This makes the export mechanism for "Extended Field Specifier Format for non-indexed MIB Object" inefficient when used for indexed MIB objects. An example is shown in Section 6.1.

When an indexed MIB object is exported in IPFIX, either the meaning of the exported value of each index may be identified or the complete OID segment identifying the instance can be sent as one piece. When the meaning of each index is identified, this index (or indices) MUST be a MIB Object Identifier (this section) or an IPFIX Information Element (see Section 5.3.4).

A MIB Object Identifier MAY be used as an index and sent as described in Figure 9. However, if a MIB Object Identifier with an index is used as an index then its indices will not be identified.

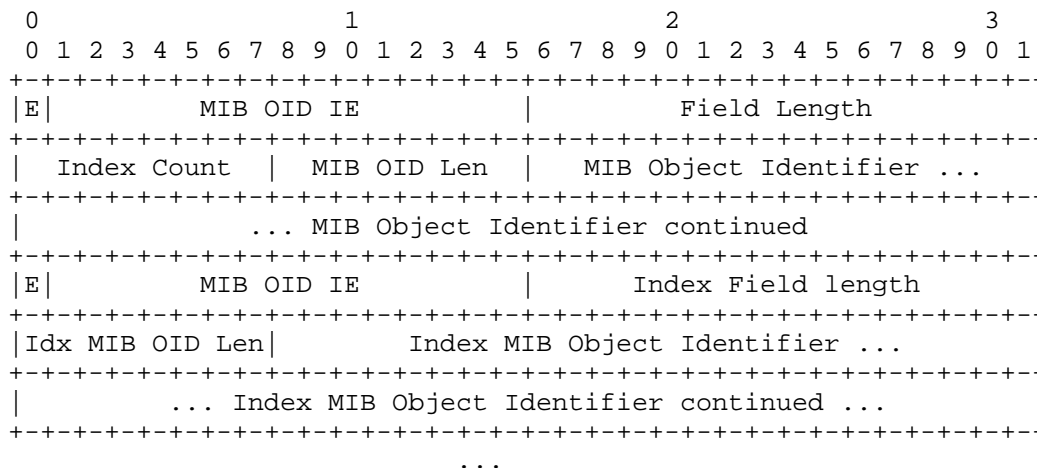


Figure 9: MIB Object Identifier Extended Field Specifier Format with a MIB Index using a normal MIB Object Identifier as index

Where:

E

Enterprise bit. This is the first bit of the Field Specifier. The value is always set to 0 for the MIB Object Identifier Extended Field Specifier Format, even if the MIB Object Identifier is enterprise-specific, because the MIB OID IE is an IANA standard field and is not enterprise-specific.

MIB OID IE

Special IPFIX Information Element, MIBObjectIdentifierMarker, that denotes that a MIB object is exported in the (Options) Template Record. When the MIB Object Identifier Information Element (MIB OID IE) is used, the MIB Object Identifier must be specified in the MIB Object Identifier Extended Field Specifier for the Collecting Process to be able to decode the Records.

Field Length

The definition is as [RFC5101]. Note that the Field Length can be expressed using reduced size encoding per [RFC5101].

Index Count

The number of indices for a MIB object, and zero for a non-indexed MIB object.

MIB Object Identifier Length

The length of the textual representation of the MIB Object Identifier that follows. This is encoded in the same manner as the variable length encoding in [RFC5101]. If the length of the MIB Object Identifier is greater than or equal to 255 octets, the length is encoded into 3 octets before the MIB Object Name Where the first octet is 255 and the length is carried in the second and third octets (as shown in Figure 8). If the MIB Object Identifier is longer than 254 characters then the length MUST be extended.

MIB Object Identifier

The textual representation of a MIB object identifier as defined in Section 4. For any indices identified using Information Elements the Enterprise bit can be 1, indicating that an Enterprise Number will follow the Information Element.

Index Field Length

The length of the encoded index field, in octets, per the Field Length definition in [RFC5101]. Note that the Index Field Length can be expressed using reduced size encoding per [RFC5101].

Index MIB Object Identifier Length

The length of the textual representation of the MIB Object Identifier being used as an index. This is encoded in the same manner as the variable length encoding in [RFC5101]. If the length of the MIB Object Identifier is greater than or equal to 255 octets, the length is encoded into 3 octets before the MIB Object Name. The first octet is 255 and the length is carried in the second and third octets.

Index MIB Object Identifier

The textual representation of a MIB object identifier as defined in Section 4.

5.3.4. Extended Field Specifier Format for an Indexed MIB Object, with an IPFIX Information Element as Index

A possible optimization for the Extended Field Specifier Format for an Indexed MIB Object as specified in Section 5.3.3 is to use an existing IPFIX Information Element, which is already present in the Flow definition, as the index for indexed MIB Object. On the top not repeating the index, the primary advantage is to make a clear link between the Flow Record values and the MIB variable index.

For example, if a Flow Record definition contains the source IP address, the destination IP address, and the ingressInterface Information Element as Flow Keys, this implies that the IP address pairs are seen on that specific interface. If the ifInOctets, indexed by that specific interface, is added to the Flow Record, it's clear from the Flow Record, that the ifInOctets is related to the same interface. If the ifInOctets was indexed by the ifIndex (as specified in Section 5.3.3), the Collector would have to hardcode that the semantic of ifIndex MIB variable is equivalent to the ingressInterface Information Element.

When an indexed MIB object is exported in IPFIX, the index (or indices) MAY be an IPFIX Information Element(s). Note that this/these IPFIX Information Element(s) MAY be an enterprise-specific Information Element.

Indexed MIB Objects, with IPFIX Information Elements as index, are exported as shown in Figure 10.

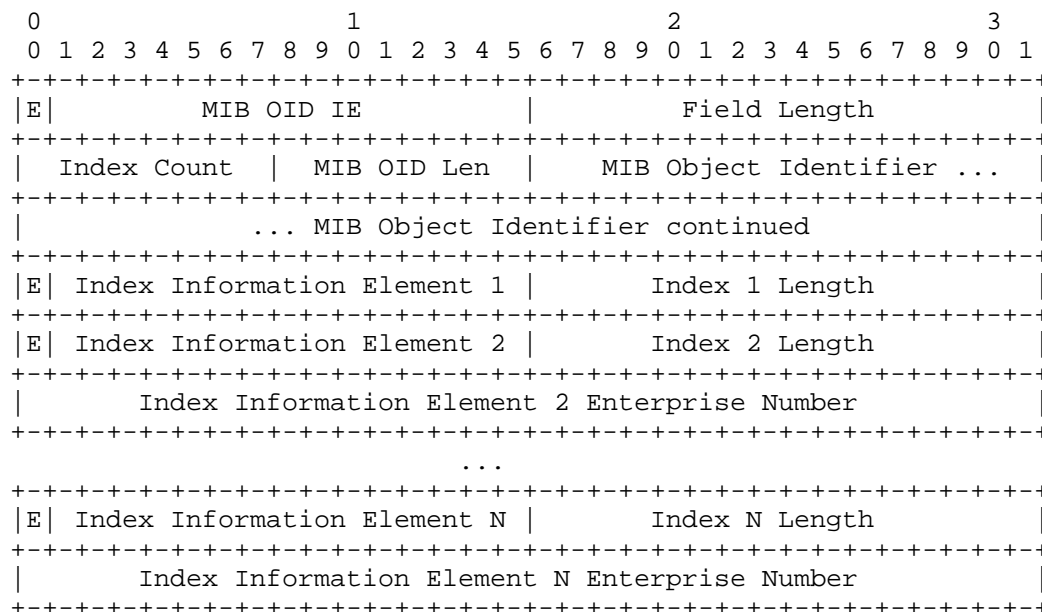


Figure 10: MIB Object Identifier Extended Field Specifier Format with an indexed MIB Object using an IPFIX Information Element as Index

Where:

E

Enterprise bit. This is the first bit of the Field Specifier. The value is always set to 0 for the MIB Object Identifier Extended Field Specifier Format, even if the MIB Object Identifier is enterprise-specific, because the MIB OID IE is an IANA standard field and is not enterprise-specific.

MIB OID IE

Special IPFIX Information Element, MIBObjectIdentifierMarker, that denotes that a MIB object is exported in the (Options) Template Record. When the MIB Object Identifier Information Element (MIB OID IE) is used, the MIB Object Identifier must be specified in the MIB Object Identifier Extended Field Specifier for the Collecting Process to be able to decode the Records.

Field Length

The definition is as [RFC5101]. The Field Length does not include the length of the index fields, since these are

specified separately. Note that the Field Length can be expressed using reduced size encoding per [RFC5101].

Index Count

The number of indices for a MIB object, and zero for a non-indexed MIB object. The index count MUST be consistent with the INDEX definition of the corresponding MIB module.

MIB Object Identifier Length

The length of the textual representation of the MIB Object Identifier that follows. This is encoded in the same manner as the variable length encoding in [RFC5101]. If the length of the MIB Object Identifier is greater than or equal to 255 octets, the length is encoded into 3 octets before the MIB Object Name where the first octet is 255 and the length is carried in the second and third octets (as shown in Figure 8). If the MIB Object Identifier is longer than 254 characters then the length MUST be extended.

MIB Object Identifier

The textual representation of a MIB object identifier as defined in Section 4.

Index Information Element 1..N

The Information Element(s) that are used as indices for the MIB Object Identifier.

Regular Information Elements, enterprise-specific Information Elements, and non-indexed MIB object identifiers may all be used as indices. However, indexed MIB object identifiers may not be used as indices because SNMP doesn't support hierarchical indexing.

Index 1..N Length

The respective index lengths for the Information Element(s) 1..N

5.3.5. Extended Field Specifier Format for an Indexed MIB Object, with one IPFIX Information Element for the OID segment identifying the instance

When MIB objects are to be exported, the Exporter may need to interact with the MIB instrumentation in an SNMP agent to obtain the

required information. For some SNMP agents, the MIB instrumentation by design does not have knowledge of the OID of the indice(s) that identify the instance of the MIB object being accessed. For example, when accessing a MIB object `ifInOctets.10`, the MIB instrumentation code may not know that the object `ifInOctets` is indexed by `ifIndex`, it is sufficient for it to map the value (10) of the `ifIndex` to an interface on the device. For such SNMP agents, the Exporter can not use the methods described in Section 5.3.3 and Section 5.3.4 without making extensive changes to the existing MIB instrumentation.

An alternate method for exporting Indexed MIB objects in such cases is to convey only the value(s) of the indice(s) that identify the instances being exported. The index count and OIDs of the indice(s) are not conveyed in the IPFIX template record. The Collecting Process is assumed to have the intelligence to understand how the exported objects are indexed. For example, it can either compile this information from the MIB Module where this object type is defined or it may be hardcoded with this information for specific MIB objects that are of interest to it. The object identifier of the indexed MIB object is split into two parts, first part is the OID prefix which is the OID of the corresponding object type and the second part is the OID segment identifying the instance. An information element called `MIBInstanceIdentifier` is defined for conveying the instance identification segment of an indexed MIB object's OID in string format. While the OID prefix is sent in the template record, the instance identifier segment is sent in the data record. Since the instance identifier segment of the MIB OID is in the data-record, the same template record can be used for exporting different instances of the same MIB object.

Indexed MIB objects, with `MIBInstanceIdentifier` as index are exported as shown in Figure 11

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|E|               MIB OID IE               |               Field Length               |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Index Count=1 | MIB OID Len | MIB Object Identifier ... |
+-----+-----+-----+-----+-----+-----+-----+-----+
|               ... MIB Object Identifier continued               |
+-----+-----+-----+-----+-----+-----+-----+-----+
|E| MIBInstanceIdentifier |MIBInstanceIdentifier Len=FFFF |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Figure 11: MIB Object Identifier Extended Field Specifier Format with an indexed MIB Object using `MIBInstanceIdentifier` as Index

Where:

E

Enterprise bit. This is the first bit of the Field Specifier. The value is always set to 0 for the MIB Object Identifier Extended Field Specifier Format, even if the MIB Object Identifier is enterprise-specific, because the MIB OID IE is an IANA standard field and is not enterprise-specific.

MIB OID IE

Special IPFIX Information Element, MIBObjectIdentifierMarker, that denotes that a MIB object is exported in the (Options) Template Record. When the MIB Object Identifier Information Element (MIB OID IE) is used, the MIB Object Identifier must be specified in the MIB Object Identifier Extended Field Specifier for the Collecting Process to be able to decode the Records.

Field Length

The definition is as [RFC5101]. The Field Length does not include the length of the index fields, since these are specified separately. Note that the Field Length can be expressed using reduced size encoding per [RFC5101].

Index Count

When the OID segment identifying the instance is exported as one string using the MIBInstanceIdentifier the Index Count value is always set to 1 to indicate that there is one information element conveying index values for this MIB object. Since the Collecting Process is assumed to know the INDEX definition of the MIB object, the actual index count need not be conveyed.

MIB Object Identifier Length

The length of the textual representation of the MIB Object Identifier that follows. This is encoded in the same manner as the variable length encoding in [RFC5101]. If the length of the MIB Object Identifier is greater than or equal to 255 octets, the length is encoded into 3 octets before the MIB Object Name where the first octet is 255 and the length is carried in the second and third octets (as shown in Figure 8). If the MIB Object Identifier is longer than 254 characters then the length MUST be extended.

MIB Object Identifier

The textual representation of a MIB object identifier as defined in Section 4.

E

Enterprise bit. This is the enterprise bit for the MIBInstanceIdentifier that follows. The value is always set to 0 when the MIBInstanceIdentifier is used because the MIBInstanceIdentifier is an IANA standard field and is not enterprise-specific.

MIB Instance Identifier

IPFIX Information Element, MIBInstanceIdentifier, that denotes that a MIB Instance identifier string is exported in the data record following the MIB Object's value. This instance identifier when concatenated with the MIB object type OID that was sent in the template record gives the complete OID of the MIB variable that is being exported.

5.4. Indices Considerations

When using an Indexed MIB Object, the Template Record contains the index/indices length. In some cases, this index/indices information might be redundant in the export information. For example, when the index is an Information Element already contained in the Template Record, the length is already part of the Template Record, and available to the Collecting Process for decode, as shown in the example in Section 6.6. A second example in Section 6.9 is when a specific MIB OID is already part of the Template Record as a standalone MIB object in a Template Record, and also reused as an index.

However, there are two cases where the index length is required. Therefore, for consistent decoding on the Collecting Process, the Index Length is always specified next to the index.

Situation 1: When a non-indexed MIB object is used as an index, and doesn't appear as a standalone MIB object in the Template Record, the Collecting Process might not want, per design, to access the MIB modules in order to find the length of the value for a particular MIB OID.

Situation 2: A Template Record might contain two similar Information Elements with different encoding lengths even if this situation is an unlikely real-world scenario), while an Indexed MIB Object might want

to refer to one of this Information Element as the index. However, without clearly specifying the index length, the Collecting Process would not know which length to decode the index with.

When an Information Element is used as index, there MUST be one and only one similar Information Element with the exact same length in the Template Record, so that the Collecting Process knows which Information Element value from the Flow Records to match. Note that this rule also implies that the reduced size encoding [RFC5101] of the Information Element in the index compared to the Information Element in the Template Record is not allowed. If the Collecting Process can not determine clearly which Information Element value to chose as the index because there are two (or more) Information Elements with the same length, then index MUST specified as the MIB Object Identifier.

An indexed MIB object MAY be indexed by a mix of MIB OID(s) and IPFIX Information Element(s)

5.5. Identifying the SNMP Context

Each MIB OID is looked up in a specific context, usually the default context. If exporting a MIB OID value that isn't in the default context then the context string MUST be identified and associated with the MIB OID. This can be done on a per template basis by exporting an Options Template Record.

A new IPFIX Information Element, "MIBObjectIdentifierMarker" has been allocated for this purpose. See Section 11.

5.6. Template Management

Templates are managed as per [RFC5101].

The Set ID field MUST contain the value TBD1 for any Template Set that contains a MIB Object Identifier Extended Field Specifier. The Template Withdrawal Message for such a Template must also use a Set ID field containing the value TBD1.

The Set ID field MUST contain the value TBD2 for any Option Template Set that contains a MIB Object Identifier Extended Field Specifier. The Template Withdrawal Message for such an Option Template must also use a Set ID field containing the value TBD2.

6. Example Use Cases

6.1. Without Using the Specifications in this Document

This example shows the need for indexed MIB objects using the example of exporting ifInOctets from Section 5.3.3.

A Template Record for exporting the ifInOctets for the interface represented by the interface ifIndex 1 (i.e., ifInOctets.1) is shown in Figure 12. While this may be useful for exporting the single ifInOctets.1 field, clearly additional Templates are required in order to export ifInOctets.2, ifInOctets.3, etc. Therefore Indexed MIB objects (per Section 5.3.3) are required in order to export arbitrary ifInOctets.x.

0																1																2																3															
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1																						
Set ID = TBD1																Length = 36																																															
Template ID = 256																Field Count = 1																																															
0 IE=MIBObjectIdentifierMarker																Field Length = 4																																															
Index Count = 0																MIB OID Len=22																MIB Object Identifier ...																															
... MIB Object Identifier = "1.3.6.1.2.1.2.2.1.10.1"																																																															
... MIB Object Identifier continued ...																																																															
... MIB Object Identifier continued ...																																																															
... MIB Object Identifier continued ...																																																															
... MIB Object Identifier continued																																																															

Figure 12: Template for exporting ifInOctets.1

6.2. Non-indexed MIB Object: Established TCP Connections

The number of established TCP connections of a remote network device could be monitored by configuring it to periodically export the number of established TCP connections to a centralized Collector. In this example, the Exporter would export an IPFIX Message every 30 minutes that contained Data Records detailing the number of established TCP connections.

The table of data that is to be exported looks like:

TIMESTAMP	ESTABLISHED TCP CONN.
StartTime + 0 seconds	10
StartTime + 60 seconds	14
StartTime + 120 seconds	19
StartTime + 180 seconds	16
StartTime + 240 seconds	23
StartTime + 300 seconds	29

Table 1: Established TCP Connections

The Template Record for such a Data Record will detail two Information Elements:

1. flowStartSeconds from [RFC5102], Information Element 150: The absolute timestamp of the first packet of this Flow.
2. tcpCurrEstab from [RFC4022], Object ID "1.3.6.1.2.1.6.9": The number of TCP connections for which the current state is either ESTABLISHED or CLOSE-WAIT.

Figure 13 shows the exported Template Set detailing the Template Record for exporting the number of established TCP connections (see Section 6.2).

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Set ID = TBD1           |           Length = 33           |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Template ID = 257       |           Field Count = 2       |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 0 | IE = flowStartSeconds         |           Field Length = 4         |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 0 | IE=MIBObjectIdentifierMarker |           Field Length = 4         |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Index Count = 0 | MIB OID Len=15 | MIB Object Identifier ... |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           ... MIB Object Identifier = "1.3.6.1.2.1.6.9"           |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           ... MIB Object Identifier continued ...                 |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           ... MIB Object Identifier continued ...                 |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           ... |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Figure 13: Example of tcpCurrEstab Template Set

Figure 14 shows the start of the Data Set for exporting the number of established TCP connections (see Section 6.2).

0										1										2										3									
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1								
										Set ID = 257																				Length = 52									
										StartTime +										0 seconds																			
																				10																			
										StartTime +										60 seconds																			
																				14																			
										StartTime +										120 seconds																			
																				19																			
										StartTime +										180 seconds																			
																				16																			
										StartTime +										240 seconds																			
																				23																			
										StartTime +										300 seconds																			
																				29																			

Figure 14: Example of tcpCurrEstab Data Set

6.3. Enterprise Specific MIB Object: Detailing CPU Load History

For the sake of demonstrating an enterprise-specific MIB object, a non-indexed MIB object is chosen for simplicity. The CPU Usage of a remote network device could be monitored by configuring it to periodically export CPU usage information, i.e. the `cpmCPUTotalMinRev` from the proprietary CISCO-PROCESS-MIB, Object ID "1.3.6.1.4.1.9.9.109.1.1.1.1.7", to a centralized Collector. In this example, the Exporter would export an IPFIX Message every 30 minutes that contained Data Records detailing the CPU 1 minute busy average at 1 minute intervals.

The table of data that is to be exported looks like:

TIMESTAMP	CPU BUSY PERCENTAGE
StartTime + 0 seconds	10%
StartTime + 60 seconds	14%
StartTime + 120 seconds	19%
StartTime + 180 seconds	16%
StartTime + 240 seconds	23%
StartTime + 300 seconds	29%

Table 2: CPU Usage Data

The Template Record for such a Data Record will detail two Information Elements:

1. flowStartSeconds from [RFC5102], Information Element 150: The absolute timestamp of the first packet of this Flow.
2. cpmCPUTotal1minRev, the overall CPU busy percentage in the last one-minute period

Figure 15 shows the exported Template Set detailing the Template Record for exporting CPU Load (see Section 6.3).

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|          Set ID = TBD1          |          Length = 47          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          Template ID = 258      |          Field Count = 2      |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 0 | IE = flowStartSeconds      |          Field Length = 4      |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 0 | IE=MIBObjectIdentifierMarker |          Field Length = 1      |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Index Count = 0 | MIB OID Len=29 |          MIB Object Identifier ... |
+-----+-----+-----+-----+-----+-----+-----+-----+
| ... MIB Object Identifier = "1.3.6.1.4.1.9.9.109.1.1.1.1.7" |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          ... MIB Object Identifier continued ...          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          ... MIB Object Identifier continued ...          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          ... MIB Object Identifier continued ...          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          ... MIB Object Identifier continued ...          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          ... MIB Object Identifier continued ...          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          ... MIB Object Identifier continued |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Figure 15: Example of CPU Load Template Set

Note that although `cpmCPUTotallminRev` is 32 bits long, reduced size encoding ([RFC5101]) has been used to encoded it within a single octet.

This example stresses that, even though the OID `cpmCPUTotallminRev` is enterprise-specific, the E bit for the `MIBObjectIdentifierMarker` is set to "0" since the "MIBObjectIdentifierMarker" Information Element is not enterprise-specific.

The corresponding Data Set does not add any value for this example, and is therefore not displayed.

6.4. Indexed MIB Object with an OID: Output Interface Queue Size in PSAMP Packet Report

Following on the example from the previous section (see Section 6.6), if the Template Record for the example Data Record does not contain the `egressInterface`, the `ifOutQLen` must be indexed by the `ifIndex`

interface index as detailed in the IF-MIB [RFC2863]:

The Template Record for the example Data Record contains the following Information Elements:

1. sourceIPv4Address
2. destinationIPv4Address
3. totalLengthIPv4
4. ifOutQLen indexed by: ifIndex

Figure 16 shows the exported Template Set detailing the Template for exporting a PSAMP Report with Interface Output Queue Length (ifOutQLen) but using the ifIndex MIB object as the exported index.

```

      0          1          2          3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|          Set ID = TBD1          |          Length = 70          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          Template ID = 259      |          Field Count = 4      |
+-----+-----+-----+-----+-----+-----+-----+-----+
|0| IE = sourceIPv4Address        |          Field Length = 4      |
+-----+-----+-----+-----+-----+-----+-----+-----+
|0| IE = destinationIPv4Address   |          Field Length = 4      |
+-----+-----+-----+-----+-----+-----+-----+-----+
|0| IE = totalLengthIPv4          |          Field Length = 4      |
+-----+-----+-----+-----+-----+-----+-----+-----+
|0|IE=MIBObjectIdentifierMarker   |          Field Length = 1      |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Index Count=1 |MIB OID Len=20 |      MIB Object Identifier ... |
+-----+-----+-----+-----+-----+-----+-----+-----+
|      ... MIB Object Identifier = "1.3.6.1.2.1.2.2.1.21"      |
+-----+-----+-----+-----+-----+-----+-----+-----+
|      ... MIB Object Identifier continued ...                  |
+-----+-----+-----+-----+-----+-----+-----+-----+
|      ... MIB Object Identifier continued ...                  |
+-----+-----+-----+-----+-----+-----+-----+-----+
|      ... MIB Object Identifier continued ...                  |
+-----+-----+-----+-----+-----+-----+-----+-----+
| ... MIB OID continued      |0|IE=MIBObjectIdentifierMarker |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1.3.6.1.2.1.2.2.1.1 length |MIB OID Len=19 | MIB Obj ID ...|
+-----+-----+-----+-----+-----+-----+-----+-----+
|      MIB Object Identifier = "1.3.6.1.2.1.2.2.1.1" ...      |
+-----+-----+-----+-----+-----+-----+-----+-----+
|      ... MIB Object Identifier continued ...                  |
+-----+-----+-----+-----+-----+-----+-----+-----+
|      ... MIB Object Identifier continued ...                  |
+-----+-----+-----+-----+-----+-----+-----+-----+
|      ... MIB Object Identifier continued ...                  |
+-----+-----+-----+-----+-----+-----+-----+-----+
| ... MIB Object Identifier cont|
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Figure 16: Example of a Template for a PSAMP Report with ifOutQLen using ifIndex from IF-MIB [RFC2863] as an index

Note that IPFIX reduced size encoding [RFC5101] has been used in this example to express ifOutQLen in a single octet, rather than the 32 bits specified in the IF-MIB [RFC2863].

The corresponding IPFIX Data Record is shown in Figure 17. For the

sake of the example, the interface index of "Eth 1/0" is 15 and the interface index of "Eth 1/1" is 16.

0										1										2										3									
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1								
Set ID = 259										Length = 72																													
										192.0.2.1																													
										192.0.2.3																													
										150																													
45										15 ...																													
...										192.0.2.4 ...																													
...										192.0.2.9 ...																													
...										350 ...																													
...										45										15 ...																			
...										192.0.2.3 ...																													
...										192.0.2.9 ...																													
...										650 ...																													
...										23										...																			
... 15																				...																			
... 192.0.2.4																				...																			
... 192.0.2.6																				...																			
... 350																				0																			
										16																													

Figure 17: Example of PSAMP Packet Report with the ifOutQLen using ifIndex from IF-MIB [RFC2863] as an index

6.5. Indexed MIB Object with Two OIDs: The ipIfStatsInForwDatagrams

MIB objects may be indexed by multiple indices. Note that all the indices apply to the MIB object, i.e. index 2 is not an index of index 1.

This example shows the export of ipIfStatsInForwDatagrams from the IP-MIB [RFC4293] indexed by the ipIfStatsIPVersion and ipIfStatsIfIndex which are provided as scope fields in an IPFIX option. Note that since these fields are used as indices for ipIfStatsInForwDatagrams, they don't need their own indices to be identified.

The Options Template Record for the example Data Record contains the following Information Elements:

1. ipIfStatsIPVersion (1.3.6.1.2.1.4.31.3.1.1) (scope field)
2. ipIfStatsIfIndex (1.3.6.1.2.1.4.31.3.1.2) (scope field)
3. ipIfStatsInForwDatagrams (1.3.6.1.2.1.4.31.3.1.12) (non-scope field) indexed by ipIfStatsIPVersion and ipIfStatsIfIndex

Figure 18 shows the exported Options Template Set.

```

      0          1          2          3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|          Set ID = TBD2          |          Length = 146          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          Template ID = 260          |          Field Count = 3          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          Scope Field Count = 2          | 0 | MIBObjectIdentifierMarker |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          Scope Field 1 Length = 1          | Index Count = 0 | MIB OID Len=22 |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          MIB Object Identifier = "1.3.6.1.2.1.4.31.3.1.1" ...          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          ... MIB Object Identifier continued ...          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          ... MIB Object Identifier continued ...          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          ... MIB Object Identifier continued ...          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          ... MIB Object Identifier continued ...          |
+-----+-----+-----+-----+-----+-----+-----+-----+
| MIB Object Identifier continued | 0 | MIBObjectIdentifierMarker |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

```

|   Scope Field 2 Length = 2   |Index Count = 0|MIB OID Len=22 |
+-----+-----+-----+-----+-----+-----+-----+-----+
|   MIB Object Identifier = "1.3.6.1.2.1.4.31.3.1.2" ...   |
+-----+-----+-----+-----+-----+-----+-----+-----+
|   ... MIB Object Identifier continued ...   |
+-----+-----+-----+-----+-----+-----+-----+-----+
|   ... MIB Object Identifier continued ...   |
+-----+-----+-----+-----+-----+-----+-----+-----+
|   ... MIB Object Identifier continued ...   |
+-----+-----+-----+-----+-----+-----+-----+-----+
|   ... MIB Object Identifier continued ...   |
+-----+-----+-----+-----+-----+-----+-----+-----+
|MIB Object Identifier continued|0| MIBObjectIdentifierMarker |
+-----+-----+-----+-----+-----+-----+-----+-----+
|   Field Length = 4   |Index Count = 2|MIB OID Len=23 |
+-----+-----+-----+-----+-----+-----+-----+-----+
|   MIB Object Identifier = "1.3.6.1.2.1.4.31.3.1.12" ...   |
+-----+-----+-----+-----+-----+-----+-----+-----+
|   ... MIB Object Identifier continued ...   |
+-----+-----+-----+-----+-----+-----+-----+-----+
|   ... MIB Object Identifier continued ...   |
+-----+-----+-----+-----+-----+-----+-----+-----+
|   ... MIB Object Identifier continued ...   |
+-----+-----+-----+-----+-----+-----+-----+-----+
|   ... MIB Object Identifier continued ...   |
+-----+-----+-----+-----+-----+-----+-----+-----+
|   ... MIB Object Identifier continued|0|MIB OID IE...|
+-----+-----+-----+-----+-----+-----+-----+-----+
|... MIB OID IE | 1.3.6.1.2.1.4.31.3.1.1 Length |MIB OID Len=22 |
+-----+-----+-----+-----+-----+-----+-----+-----+
|   MIB Object Identifier = "1.3.6.1.2.1.4.31.3.1.1" ...   |
+-----+-----+-----+-----+-----+-----+-----+-----+
|   ... MIB Object Identifier continued ...   |
+-----+-----+-----+-----+-----+-----+-----+-----+
|   ... MIB Object Identifier continued ...   |
+-----+-----+-----+-----+-----+-----+-----+-----+
|   ... MIB Object Identifier continued ...   |
+-----+-----+-----+-----+-----+-----+-----+-----+
|   ... MIB Object Identifier continued ...   |
+-----+-----+-----+-----+-----+-----+-----+-----+
|   ... MIB Object Identifier   |0|           MIB OID IE   |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1.3.6.1.2.1.4.31.3.1.2 Length | MIB OID Len=22| MIB Obj ID ...|
+-----+-----+-----+-----+-----+-----+-----+-----+
|   MIB Object Identifier = "1.3.6.1.2.1.4.31.3.1.2" ...   |
+-----+-----+-----+-----+-----+-----+-----+-----+
|   ... MIB Object Identifier continued ...   |
+-----+-----+-----+-----+-----+-----+-----+-----+

```



```

|          ... MIB Object Identifier continued ...          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          ... MIB Object Identifier continued ...          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|    ... MIB Object Identifier    |
+-----+-----+-----+-----+-----+-----+-----+

```

Figure 18: Example of an Options Template for an Indexed MIB Object with two indices.

6.6. Indexed MIB Object with an IPFIX Information Element: Output Interface Queue Size in PSAMP Packet Report

If a PSAMP Packet Report [RFC5476] was generated on any dropped packets on an interface then it may be desirable to know if the send queue on the output interface was full. This could be done by exporting the size of the send queue (ifOutQLen) in the same Data Record as the PSAMP Packet Report.

The exported data looks like:

SRC ADDR	DST ADDR	PAK LEN	OUTPUT I/F	OUTPUT Q. LEN (ifOutQLen)
192.0.2.1	192.0.2.3	150	Eth 1/0 (15)	45
192.0.2.4	192.0.2.9	350	Eth 1/0 (15)	45
192.0.2.3	192.0.2.9	650	Eth 1/0 (15)	23
192.0.2.4	192.0.2.6	350	Eth 1/1 (16)	0

Table 3: Packet Report with Interface Output Queue Length (ifOutQLen) Data

The MIB object for the Interface Output Queue Length, ifOutQLen ("1.3.6.1.2.1.2.2.1.21"), is indexed by the ifIndex interface index as detailed in the IF-MIB [RFC2863]. If, for example, the interface index of "Eth 1/0" in the example is 15, the full MIB Object Identifier for (ifOutQLen) would be "1.3.6.1.2.1.2.2.1.21.15". Without a method to specify the index the full MIB OID would have to be used, which would mean specifying a new Template Record. Rather than export a separate Template Record for each Interface Index, it is more practical to identify the index in the Data Record itself.

In fact, only how the indexed object was indexed is necessary, although it is often useful to specify the index value. The example identifies the Egress Interface, but for other uses it may be sufficient to know that the ifOutQLen value was taken for the interface that the packet was switched out of, without identifying the actual interface.

The Template Record for the example Data Record contains the following Information Elements:

1. sourceIPv4Address
2. destinationIPv4Address
3. totalLengthIPv4
4. egressInterface
5. ifOutQLen indexed by: egressInterface

Figure 19 shows the exported Template Set detailing the Template for exporting a PSAMP Report with Interface Output Queue Length (ifOutQLen) (see Section 6.4).

```

0          1          2          3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|          Set ID = TBD1          |          Length = 54          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          Template ID = 261      |          Field Count = 5      |
+-----+-----+-----+-----+-----+-----+-----+-----+
|0|    IE = sourceIPv4Address    |          Field Length = 4      |
+-----+-----+-----+-----+-----+-----+-----+-----+
|0|    IE = destinationIPv4Address |          Field Length = 4      |
+-----+-----+-----+-----+-----+-----+-----+-----+
|0|    IE = totalLengthIPv4      |          Field Length = 4      |
+-----+-----+-----+-----+-----+-----+-----+-----+
|0|    IE = egressInterface      |          Field Length = 4      |
+-----+-----+-----+-----+-----+-----+-----+-----+
|0|IE=MIBObjectIdentifierMarker |          Field Length 4      |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Index Count=1 |MIB OID Len=20 |          MIB Object Identifier ... |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          ... MIB Object Identifier = "1.3.6.1.2.1.2.2.1.21"      |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          ... MIB Object Identifier continued ...                  |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          ... MIB Object Identifier continued ...                  |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          ... MIB Object Identifier continued ...                  |
+-----+-----+-----+-----+-----+-----+-----+-----+
| ... MIB OID continued          |0|    IE = egressInterface      |
+-----+-----+-----+-----+-----+-----+-----+-----+
| egressInterface Length = 4      |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Figure 19: Example of Template for a PSAMP Report with ifOutQLen indexed by egressInterface

The corresponding IPFIX Data Record is shown in Figure 20. For the sake of the example, the interface index of "Eth 1/0" is 15 and the interface index of "Eth 1/1" is 16.

0										1										2										3									
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1								
Set ID = 261										Length = 84																													
										192.0.2.1																													
										192.0.2.3																													
										150																													
										15 (Eth 1/0)																													
										45																													
										192.0.2.4																													
										192.0.2.9																													
										350																													
										15 (Eth 1/0)																													
										45																													
										192.0.2.3																													
										192.0.2.9																													
										650																													
										15 (Eth 1/0)																													
										23																													
										192.0.2.4																													
										192.0.2.6																													
										350																													
										16 (Eth 1/1)																													
										0																													

Figure 20: Example of PSAMP Packet Report with ifOutQLen indexed by egressInterface

6.7. Indexed MIB Objects with a mix of MIB OID and IPFIX Information Element

TODO.

6.8. Indexed MIB Object with MIBInstanceIdentifier Information Element: ipIfStatsOutOctets

This example shows the export of ipIfStatsOutOctets from the IP-MIB [RFC4293] indexed by the ipIfStatsIPVersion and ipIfStatsIfIndex, using the MIBInstanceIdentifier Information Element to carry the index information.

The exported data looks like:

ipIfStatsIPVersion	ipIfStatsIfIndex	ipIfStatsOutOctets
1(IPv4)	10	235876
2(IPv6)	11	38688

Table 4: The number octets in IP datagrams delivered to the lower layers for transmission

The MIB object ipIfStatsOutOctets ("1.3.6.1.2.1.4.31.3.1.32"), is indexed by ipIfStatsIPVersion and ipIfStatsIfIndex as detailed in IP-MIB [RFC4293]. The instance of the ipIfStatsOutOctets for the IPv4 protocol on the interface identified by ifIndex 10 is identified in the data record with the instance identifier segment ("1.10") in string format, while the instance of the ipIfStatsOutOctets for the IPv6 protocol on the interface identified by ifIndex 11 is identified in the data record with the instance identifier segment ("2.11") in string format.

The Template Record for the example Data Records contains the following Information Elements:

1. ipIfStatsOutOctets (1.3.6.1.2.1.4.31.3.1.32)

Figure 21 shows the exported Template Set.

```

      0          1          2          2
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|          Set ID = TBD1          |          Length = 86          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          Template ID = 264      |          Field Count = 1      |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 0 | IE=MIBObjectIdentifierMarker |          Field Length = 4    |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Index Count=1 | MIB OID Len=23 |          MIB Object Identifier ... |
+-----+-----+-----+-----+-----+-----+-----+-----+
| ...MIB Object Identifier = "1.3.6.1.2.1.4.31.3.1.32" |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          ... MIB Object Identifier continued ...          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          ... MIB Object Identifier continued ...          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          ... MIB Object Identifier continued ...          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          ... MIB Object Identifier continued ...          |
+-----+-----+-----+-----+-----+-----+-----+-----+
| ...MIB OID          | 0 | MIBInstanceIdentifier IE          | FieldLength... |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          ... = FFFF          |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Figure 21: Example of a Template for an MIB Objects that use the MIBInstanceIdentifier Information Element

The corresponding IPFIX Data Record is shown in Figure 22.

Variable length encoding is used for MIBInstanceIdentifier Information Element.

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|               Set ID = 264               |               Length = 22               |
+-----+-----+-----+-----+-----+-----+-----+-----+
|               ipIfStatsOutOctets = 235876               |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Length = 4 |               "1.10"...               |
+-----+-----+-----+-----+-----+-----+-----+-----+
| ... |               ipIfStatsOutOctets = 38688               |
+-----+-----+-----+-----+-----+-----+-----+-----+
| ... | Length = 4 |               "2.11"...               |
+-----+-----+-----+-----+-----+-----+-----+-----+
| ... |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Figure 22: Example of ipIfStatsOutOctets using ipIfStatsIPVersion and ipIfStatsIfIndex as indices

6.9. Using MIB Objects as IPFIX Options Scope fields

Both indexed and non-indexed MIB Objects may be used as IPFIX Options Scope fields as discussed in Section 5.2.

6.9.1. Using non-Indexed MIB Objects as Option Scope fields

In this example, a Cisco Telepresence system uses an IPFIX option to report bandwidth usage statistics. The ctpcLocalAddrType and ctpcLocalAddr OIDs from the CISCO-TELEPRESENCE-CALL MIB are used as scope fields to identify the Telepresence system. The ctpcLocalAddrType is expressed with a fixed size of 1 octet, while the ctpcLocalAddr is expressed using a variable length field.

These scope fields are followed by two non-scope fields containing the number of packets and bytes. IPFIX reduced size encoding is used to express each of these fields in 32 bits.

Therefore the Options Template Record for the example Data Record contains the following Information Elements:

1. ctpcLocalAddrType (1.3.6.1.4.1.9.9.644.1.2.1) (scope field)
2. ctpcLocalAddr (1.3.6.1.4.1.9.9.644.1.2.2) (scope field)
3. octetDeltaCount (non-scope field)
4. packetDeltaCount (non-scope field)

The IPFIX Options Template Record is shown in Figure 23.

```

      0          1          2          3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|          Set ID = TBD2          |          Length = 80          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          Template ID = 262      |          Field Count = 4      |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          Scope Field Count = 2  | 0 | MIBObjectIdentifierMarker |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          Scope Field 1 Length = 1 | Index Count = 0 | MIB OID Len=25 |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          MIB Object Identifier = "1.3.6.1.4.1.9.9.644.1.2.1" ... |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          ... MIB Object Identifier continued ...                |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          ... MIB Object Identifier continued ...                |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          ... MIB Object Identifier continued ...                |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          ... MIB Object Identifier continued ...                |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          ... MIB Object Identifier continued ...                |
+-----+-----+-----+-----+-----+-----+-----+-----+
| ... MIB OID ID | 0 | MIBObjectIdentifierMarker | Scope Field ... |
+-----+-----+-----+-----+-----+-----+-----+-----+
| ...Length=65535 | Index Count = 0 | MIB OID Len=25 | MIB OID ID ... |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          ... MIB Object Identifier = "1.3.6.1.4.1.9.9.644.1.2.2" ... |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          ... MIB Object Identifier continued ...                |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          ... MIB Object Identifier continued ...                |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          ... MIB Object Identifier continued ...                |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          ... MIB Object Identifier continued ...                |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 0 |          octetDeltaCount = 1          |          Field Length = 4          |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 0 |          packetDeltaCount = 2          |          Field Length = 4          |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Figure 23: Example of an IPFIX Options Template Record using non-Indexed MIB Objects as scope fields

The corresponding IPFIX Options Data Record is shown in Figure 24.

```

      0          1          2          3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Set ID = 262           |           Length = 18           |
+-----+-----+-----+-----+-----+-----+-----+-----+
| AddrType = 1 | Length = 4 | ctpcLocalAddrSystemID = ... |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           ... 192.0.2.1           |   octetDeltaCount = nnnn ...   |
+-----+-----+-----+-----+-----+-----+-----+-----+
| ... octetDeltaCount continued | packetDeltaCount = nnnn ... |
+-----+-----+-----+-----+-----+-----+-----+-----+
| ... packetDeltaCount continued |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Figure 24: Example of an IPFIX Options Data Record using non-Indexed MIB Objects as scope fields

6.9.2. Using Indexed MIB Objects as Option Scope fields

In this example, interface statistics are reported using ifName and ifInOctets from the IF-MIB [RFC2863]. Both of these fields are indexed by the ifIndex. The ifName and ifIndex are scope fields.

Therefore the Options Template Record for the example Data Record contains the following Information Elements:

1. ifName (1.3.6.1.2.1.31.1.1.1.1) (scope field) indexed by ifIndex
2. ifIndex (1.3.6.1.2.1.2.2.1.1) (scope field)
3. ifInOctets (1.3.6.1.2.1.2.2.1.10) (non-scope field) indexed by ifIndex

The IPFIX Options Template Record is shown in Figure 25.

```

      0          1          2          3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Set ID = TBD2           |           Length = 137           |
+-----+-----+-----+-----+-----+-----+-----+-----+
|           Template ID 263           |           Field Count = 3           |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Scope Field Count = 2 | 0 | MIBObjectIdentifierMarker |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Scope Field 1 Length = 65535 | Index Count = 1 | MIB OID Len=22 |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

```
| MIB Object Identifier = "1.3.6.1.2.1.31.1.1.1.1" |
|-----|
| ... MIB Object Identifier continued ... |
|-----|
| ... MIB Object Identifier continued ... |
|-----|
| ... MIB Object Identifier continued ... |
|-----|
| ... MIB Object Identifier continued ... |
|-----|
|MIB Object Identifier continued|0| MIBObjectIdentifierMarker |
|-----|
| Scope Field 1 index Length = 4|MIB OID Len=19 | MIB OID ID ...|
|-----|
| ... MIB Object Identifier = "1.3.6.1.2.1.2.2.1.1" ... |
|-----|
| ... MIB Object Identifier continued ... |
|-----|
| ... MIB Object Identifier continued ... |
|-----|
| ... MIB Object Identifier continued ... |
|-----|
| ... MIB Object Identifier continued ... |
|-----|
| ... MIB Object Identifier continued ... |
|-----|
| ... MIB Obj Identifier continued |0| MIBObject...|
|-----|
|...Ident Marker| Field Length = 4 |Index Count = 1|
|-----|
|MIB OID Len=20 |MIB Object Identifier="1.3.6.1.2.1.2.2.1.10"...|
|-----|
| ... MIB Object Identifier continued ... |
|-----|
| ... MIB Object Identifier continued ... |
|-----|
| ... MIB Object Identifier continued ... |
|-----|
| ... MIB Object Identifier continued ... |
```

```

| ... MIB OID |0| MIBObjectIdentifierMarker | Field 1 ... |
+-----+
|... index Len=4|MIB OID Len=19 | MIB Object Identifier ... |
+-----+
| ... MIB Object Identifier = "1.3.6.1.2.1.2.2.1.1" ... |
+-----+
| ... MIB Object Identifier continued ... |
+-----+
| ... MIB Object Identifier continued ... |
+-----+
| ... MIB Object Identifier continued ... |
+-----+
|... MIB Obj Id |
+-----+

```

Figure 25: Example of an IPFIX Options Template Record using Indexed MIB Objects as scope fields

The corresponding IPFIX Options Data Record is shown in Figure 26. For the sake of the example, the interface index of "Eth 1/1" is 15 and the ifInOctets are 1000.

```

0          1          2          3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+
|          Set ID = 263          |          Length = 20          |
+-----+
| Length = 7 |          ifName = "Eth 1/1" ... |
+-----+
|          ... ifName continued          |
+-----+
|          ifIndex = 15          |
+-----+
|          ifInOctets = 1000          |
+-----+

```

Figure 26: Example of an IPFIX Options Data Record using Indexed MIB Objects as scope fields

6.10. Using MIB Objects with IPFIX Structured Data

It's possible to export both indexed and non-indexed MIB Objects using IPFIX Structured Data per [RFC6313] as shown in the example below.

TODO: insert example.

7. Configuration Considerations

When configuring a MIB OID for export, consideration should be given to whether the SNMP Context String should also be configurable. If a non-default Context String is used then it should be associated with the fields as per Section 5.5.

8. The Collecting Process's Side

This section describes the Collecting Process when using SCTP and PR-SCTP as the transport protocol. Any necessary changes to the Collecting Process specifically related to TCP or UDP transport protocols are specified in section 10 of [RFC5101].

The specifications in section 9 of [RFC5101] also apply to Collector's that implement this specification. In addition, the following specifications should be noted.

A Collecting Process that implements this specification **MUST** be able to receive Set IDs TBD1 and TBD2, as specified in this document.

A Collecting Process that implements this specification **MUST** have access to MIB modules in order to look up the received MIB Object Identifiers and find the type and name of MIB OID fields used in received templates. It should be noted that since reduced size encoding **MAY** be used by the Exporting Process then the Collecting Process cannot assume a received size for a field is the maximum size it should expect for that field.

If a Collecting Process receives a MIB Object ID that it cannot decode, it **SHOULD** log an error.

If a Collecting Process receives a MIB Object ID for an indexed MIB Object but isn't sent the appropriate number of indices then it **SHOULD** log an error, but it **MAY** use the Template Record to decode the Data Records as the associated indices are purely semantic information.

9. Applicability

Making available the many and varied items from MIB modules opens up a wide range of possible applications for the IPFIX protocol, some quite different from the usual flow information. Some potential enhancements for traditional applications are detailed below:

Some monitoring applications periodically export an interface id to

interface name mapping using IPFIX Options Templates. This could be expanded to include the MIB object "ifInUcastPkts" of the IF-MIB [RFC2863] indexed using the ingressInterface Information Element, as an index. This would give the input statistics for each interface which can be compared to the flow information to ensure the sampling rate is expected. Or, if there is no sampling, to ensure that all the expected packets are being monitored.

10. Security Considerations

For this extension to the IPFIX protocol, the same security considerations as for the IPFIX protocol apply [RFC5101].

The access to MIB objects is controlled by the configuration of the IPFIX exporter. This is consistent with the way IPFIX controls access to other Information Elements in general. The configuration of an IPFIX exporter determines which MIB objects are included in IPFIX flow records sent to certain collectors. Network operators should take care that only MIB objects are included in IPFIX flow records that the receiving flow collector is allowed to receive.

11. IANA Considerations

11.1. New Set IDs

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

The previously reserved Set ID values of TBD1 and TBD2 are allocated in IANA's IPFIX Set IDs registry [IANA-SETS], and are used as specified in this document. All other Set ID values are reserved for future use. Set ID values above 255 are used for Data Sets.

11.2. New Data Types

A new mibObject data type must be allocated in IANA's IPFIX Information Element Data Types registry, [IANA-DATATYPES].

11.3. New Information Elements

Two new Information Elements, "MIBObjectIdentifierMarker", and "MIBInstanceIdentifier" must be allocated in IANA's IPFIX registry, [IANA-IPFIX]:

MIB Object Identifier Marker

Description: An IPFIX Information Element ("MIBObjectIdentifierMarker") that denotes that a MIB Object Identifier is exported in the (Options) Template Record.

Abstract Data Type: mibObject

Data Type Semantics: identifier

ElementId: TBD

Status: current

Reference: [this document].

MIB Instance Identifier

Description: IPFIX Information Element, MIBInstanceIdentifier, that denotes that a MIB Instance identifier string is exported in the data record following the MIB Object's value. This instance identifier when concatenated with the MIB object type OID that was sent in the template record gives the complete OID of the MIB variable that is being exported.

Abstract Data Type: mibObject

Data Type Semantics: identifier

ElementId: TBD

Status: current

Reference: [this document].

12. Acknowledgements

The authors would like to thank Andrew Johnson for his collaboration on the first version of the draft.

13. References

13.1. Normative References

[IANA-DATATYPES]

IANA, "IPFIX Information Element Data Types registry", <ht

[tp://www.iana.org/assignments/ipfix/
ipfix.xml#ipfix-information-element-data-types](http://www.iana.org/assignments/ipfix/ipfix.xml#ipfix-information-element-data-types)>.

[IANA-IPFIX]

IANA, "IPFIX Information Elements registry",
<<http://www.iana.org/assignments/ipfix/ipfix.xml>>.

[IANA-SETS]

IANA, "IPFIX Set IDs registry", <<http://www.iana.org/assignments/ipfix/ipfix.xml#ipfix-set-ids>>.

[PEN]

IANA, "Private Enterprise Numbers registry",
<<http://www.iana.org/assignments/enterprise-numbers>>.

[RFC2119]

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

[RFC2578]

McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.

[RFC2863]

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

[RFC4293]

Routhier, S., "Management Information Base for the Internet Protocol (IP)", RFC 4293, April 2006.

[RFC5101]

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

[RFC5102]

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

13.2. Informative References

[RFC2982]

Kavasseri, R., "Distributed Management Expression MIB", RFC 2982, October 2000.

[RFC3444]

Pras, A. and J. Schoenwaelder, "On the Difference between Information Models and Data Models", RFC 3444, January 2003.

[RFC4022]

Raghunarayan, R., "Management Information Base for the Transmission Control Protocol (TCP)", RFC 4022, March 2005.

[RFC5476] Claise, B., Johnson, A., and J. Quittek, "Packet Sampling (PSAMP) Protocol Specifications", RFC 5476, March 2009.

[RFC6313] Claise, B., Dhandapani, G., Aitken, P., and S. Yates, "Export of Structured Data in IP Flow Information Export (IPFIX)", RFC 6313, July 2011.

Authors' Addresses

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

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

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

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

Srikar
Cisco Systems, Inc.
Mail Stop BGL13/3/, SEZ Unit, Cessna Business Park, Kadubeesanahalli
Village Varthur Hobli, Sarjapur Marathalli Outer Ring Road
Bangalore, KARNATAKA 560 103
IN

Phone: +91 80 4426 3264
Email: srikar@cisco.com

Juergen Schoenwaelder
Jacobs University Bremen
Campus Ring 1
Bremen, 28725
Germany

Phone: +49 421 200-3587
Email: j.schoenwaelder@jacobs-university.de

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

