

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

Internet Draft

PENIE

November 9, 2012

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

Internet Draft

PENIE

November 9, 2012

Table of Contents

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

1. Introduction

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

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

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

2. Options Record Format

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

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

Inacio

Expires May 13, 2013

[Page 4]

Internet Draft

PENIE

November 9, 2012

[illegible]

Figure 1: Example PENIE Template Definition

[illegible]

```

+                                     +
|                                     |
+                                     +
|                                     |
+                                     +
|                                     |
+                                     +
|                                     |
+                                     +
|                                     |
+                                     +
|                                     |
+                                     +
|                                     |
+-----+-----+-----+-----+
|                                     |
+-----+-----+-----+-----+

```

Figure 2: Example PENIE Data Record

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

[3. Registry Design](#)

[3.1. Registry Introduction](#)

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

[3.2. Registry Informational Elements](#)

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

- o Registry ID - This is an ID that can be used by the creator of the registry to be able to track the registry as a unique item.

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

Each information element defined in the registry are as follows:

- o ID - The information element ID.
- o PEN - The private enterprise number.
- o Data type - The data type of the element, as defined in [RFC 5610](#).
- o Semantics - The semantic of the element, as defined in [RFC 5610](#).

- o Units - The units of the element, as defined in [RFC 5610](#).
- o Description - The human readable (and hopefully understandable) description of the element.
- o MIME Path - An optional MIME path definition of the element.

[3.3](#). Registry Formatting

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

5. Security Considerations

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

[6](#). IANA Considerations

IANA needs to create two registries with expert review:

Security types Policy types

and create a code point for a new information element.

[6](#). References

[6.1](#). Normative References

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

- [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

```
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" maxIncl
```

```
    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.

Inacio

Expires May 13, 2013

[Page 10]

Internet Draft

PENIE

November 9, 2012

.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

Inacio

Expires May 13, 2013

[Page 11]

Internet Draft

PENIE

November 9, 2012

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
```