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.

Table of Contents

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 readabilty.]

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 unprefixed 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 natually represented by the presence or absence of a value in the structure of the document in the Enclosing Context.]

```
boolean-yes = "1" / "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)

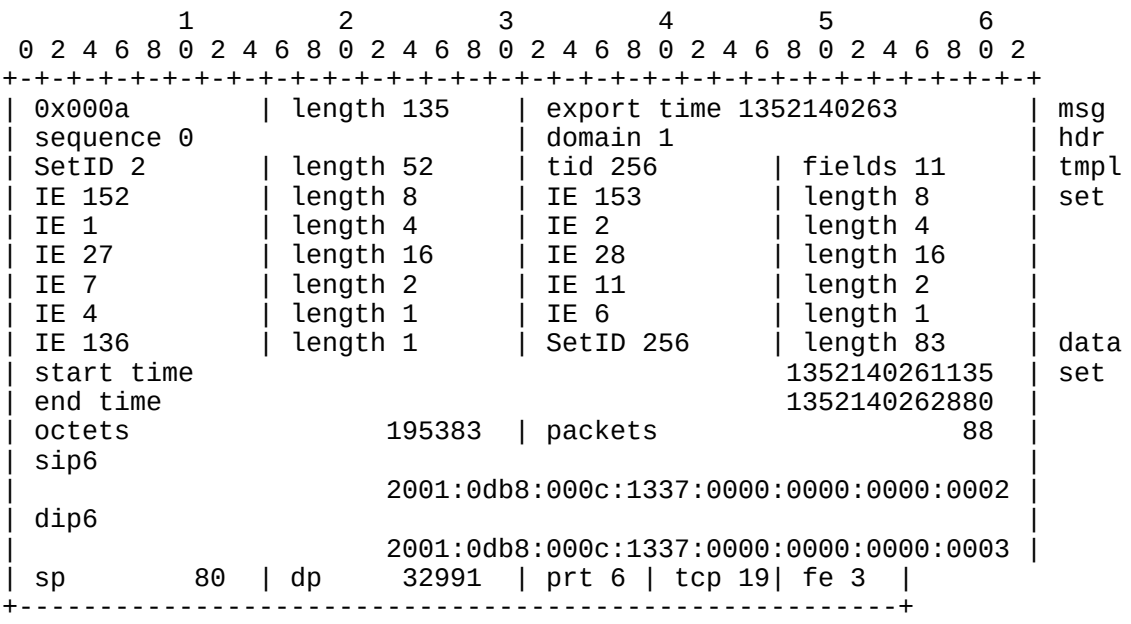

Figure 2: IPFIX message containing sample flow

```
{
    "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