Applying RFC 6313:
Comments on Structured Data and the Semantics of IPFIX Records
by a frustrated collector implementor

B. Trammell

(with thanks to H. Kaplan, who brought this up a while ago…)

IETF 82 – Taipei, Taiwan, 17 November 2011

IPFIX per 5101 (-bis)

- **Types** specify data encoding.
  - `typedef uint32_t ip4addr_t; // big-endian`
- ***Information elements*** **(IEs)** specify the semantic meaning of a specific field of a record.
  - `ipaddr_t sourceIPv4Address;`
- ***Templates*** are ordered lists of IEs specifying the structure of a record.
  - `struct flow_st { …`
- Easy to understand, easy to implement…

From Templates to Structures

- Example: *twoflow*
  - flow by address pair for capacity planning purposes
-
```
typedef uint32_t time_t;    // epoch sec
typedef uint32_t ip4addr_t; // big endian
```

-
```
struct twoflow_st {
    time_t    flowStartSeconds;
    time_t    flowEndSeconds;
    ip4addr_t sourceIPv4Address;
    ip4addr_t destinationIPv4Address;
    uint64_t  deltaPacketCount;
}
```

Problems with 5101 (-bis)

- Templates are great if all your records look the same.
- But…
  - Poor handling of multiplicity
  - Poor handling of subordinate structures
  - Poor handling of type alternation
    - e.g. identification of an link by interface/prefix/MAC/etc.
- Template explosion for certain applications
  - e.g. packet/flow decode in DPI
- Structured data (RFC 6313) to the rescue!

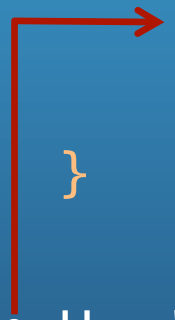
What's new in 6313

- New **basicList** type
  - variable length array of a single IE
- New **subTemplateList** type
  - variable-length array of a subordinate structure
- New **subTemplateMultiList** type
  - variable-length array of subordinate structures of varying types
  - Or: an entire IPFIX Message Body embedded within a single Data Record.
- Powerful **semantics** attached to each of these types
  - `oneOf, oneOrMoreOf, allOf, noneOf, ordered`
- New **generic IEs** for each of these new types

Applying 6313

- Example: aggregate *twoflows* by disjoint sets of source IP addresses
  - Replace `sourceIPv4Address` with a `basicList` that contains `sourceIPv4Address` and `oneOf` semantics.

```
struct twoflow_st {
    time_t       flowStartSeconds;
    time_t       flowEndSeconds;
    basicList_t  basicList;
    ip4addr_t    destinationIPv4Address;
    uint64_t     deltaPacketCount;
}
```



- Here's where the problems begin.

IPFIX per 6313

- ***Information elements*** **(IEs)** specify the semantic meaning of a specific field of a record.
  - ***Unless they are generic.***
  - Semantic meaning of generic IEs determined by content.
- ***Templates*** are ordered lists of IEs specifying the structure of a record.
  - ***Unless they contain generic IEs.***
  - Record structure information with generic IEs unavailable until record parse is completed.
- Decisions about record type and structure ***must be deferred*** to record parse time with 6313 generic IEs.

Why is this bad?

- Muddles IPFIX self-description: instead of templates describing data, now data describes data too.
- Demux on Template ID at collector impossible
- Record validation severely complicated
  - What does a *twoflow* collector do with a record containing a `basicList` of `octetDeltaCount`?
  - I don't know either — but it has to parse the whole record to decide.

What I really, really want…

```
struct twoflow_st {
     time_t                    flowStartSeconds;
     time_t                    flowEndSeconds;
     std::vector<ipv4addr_t>   sourceIPv4Addresses;
     ip4addr_t                 destinationIPv4Address;
     uint64_t                  deltaPacketCount;
}
```

- Separate structure from encoding
- Ability to label/differentiate structured data IEs in a template
- (And I want all this for free without burning new SetIDs)

Solutions

- Solutions? I'm just here to complain…
- Non-generic `basicList` and `subTemplateList` IEs might help
  - e.g. `sourceAddressList`
  - + enables demux on template ID
  - + makes Structured Data properly self-describing again
  - - leads to IE explosion, which 6313 meant to avoid
  - - introduces new runtime constraints at the collector
    - + which exist in reality anyway
    - need to define representation for allowable list contents
- `subTemplateMultiList` is an entirely separate beast.