IPFIX Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 19, 2014

P. Aitken, Ed.
B. Claise
S. B S
C. McDowall
Cisco Systems, Inc.
J. Schoenwaelder
Jacobs University Bremen
February 15, 2014

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

Abstract

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

An IPFIX Option Template and method are specified, which are used to export the extra information required to fully describe Simple Network Management Protocol (SNMP) MIB Objects.

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 19, 2014.

Table of Contents

1. Open Issues / To do list

- "timestamps, exporters, and other animals" -> see the mailing list.

- Some TODOs in the XML version:

  - write section 6.7: "Indexed MIB Objects with a mix of MIB OID and IPFIX Information Element"

  - write section 6.10: "Using MIB Objects with IPFIX Structured Data"

  - write section 6.11: "Using IPFIX Structured Data to group the index MIB and indices"

- RFC 5610: explain what needs to be updated.

- ID to name mappings? -> use this for an example in section 5.

- What does this mean? : "(Consider the counter synchronization issue, non-key info should be static)".

- (JS) Inacio's figure: send email to the mailing list.

- Tidy up the XML.

- Add missing examples in sections 6.7, 6.9.1, 6.9.2, 6.10, 6.11.

- Describe all the orphaned figures ("Figure NN shows ...").

- Add example which uses mibContextEngineID and mibContextName.

- Clear up figure 6/7 or surrounding text - "I have trouble to understand Figure 6 and Figure 7." js

- "I am confused by Figure 14(-03 draft) with a "Set ID = ??" Is this left from the earlier version?" - AF

- Make clear ordering constraints or lack of "What about ordering requirements? I assume no lexicographic ordering needed when MIB data is moved via IPFIX?" - JS

- "In the enumeration in section 5.2 (-03 draft), I suggest to inline the explanations rather than having the explanation below." - JS

- Generalise the enterprise specific example, then probably remove the example: "Is section 6.3 really needed? The only important sentence is this: This example stresses that, even though the OID cpmCPUTotal1minRev is enterprise-specific, the E bit for the mibObjectValue and mibObjectIdentifier is set to "0" since the "mibObjectValue" and "mibObjectIdentifier" Information Element is not enterprise-specific. This should be generalized and written down where the normative text is of the mibObjectValue and mibObjectIdentifier definitions, e.g. The E bit of the mibObjectValue or mibObjectIdentifier Information Elements is set to "0" since they are not enterprise-specific. This holds true even if the data carried inside the mibObjectValue or mibObjectIdentifier may be enterprise specific.H18" - JS

- "Should ipIfStatsIPVersion not be InetVersion in Figure 24?" - JS

- "Is Figure 22(-03 draft) a "Options Template Set"? I expected this to be a Template Set" - JS

- "Am I correct that the example in 6.4 (-03 draft) is supposed to show the value of ipIfStatsInForwDatagrams (10000 and 20000) for ip4 and ip6 on interface 10? If so, note that InetVersion encodes IPv4 using '1' and IPv6 using '2'. In any case, showing how a data record is encoded is very useful but requires some textual explanation." - JS

- "Why do we talk about IPFIX transports in section 8(-03 draft)? All this should be pretty agnostic to the choice of the transport, no?" - JS

- Clear up relationship to SNMP - Ie Accessing MIB not accessing via / through SNMP.

- "I think there needs to be some more explicit text saying that the security administrator must make sure that IPFIX export of MIB objects does not cause a hole into the SNMP access control configuration. (Some SNMP hardliners might even say that IPFIX should access data through the SNMP access control subsystem, in which case there would need to be proper parameters such as a (securityName, securityModel, securityLevel) tuple..." - JS

- Check all uses of 'OID' for correctness - "Section 11.2 says "the MIB OID" which is ambiguous at best. It should probably be "the OID of a MIB object" or even better "the OID assigned to the MIB object definition"." - JS

- Spell out 1-1 mapping of fields to mibFieldOptions: "Figure 4 helps getting an overview (but perhaps a bit late). But you seem to make the assumption that there is a fixed # of mibObjectValues mapping to the same fixed number of mibFieldOptions. This may be fine - I am just noting this (and if correct it might be good to spell it out)."

- Add note about 7011 Section 8 "Options Templates and Templates that are related or interdependent (e.g., by sharing common properties as described in [RFC5473]) SHOULD be sent together in the same IPFIX Message" - CMCD

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 [RFC7012], Information Elements coming from the SNMP world have already been specified, e.g., ingressInterface and egressInterface both refer to the ifIndex defined in [RFC2863].

In particular the Management Information Base was designed as a seperate system of definitions. There are no dependencies between the SMIv2 [RFC2578] and the SNMP protocol. This opens up the possibility to exporting Objects defined via the MIB over other protocols.

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 Data Record. This way, the duplication of data models [RFC3444], both as SMIv2 MIB objects and IPFIX Information Elements, out of the same information model [RFC3444] would be avoided.

This document defines three classes of new IPFIX Information Elements. These are used to export values from the MIB, export required Object Identifier information and optionally export type data from a MIB Module.

- mibObjectValue<> IEs
- mibFieldOption IEs
- mibTypeInformation IEs

This document also defines some standard Option Templates that are used as part of the mechanism to export MIB Object meta data.

- mibFieldOption
- mibSequenceOption
- mibTypeOption

This document specifies a method for creating an IPFIX Option Template which is used to export the extra data required to describe MIB variables (see Section 5.1.2). This allows IPFIX 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). Enterprise-specific MIB Object Identifiers are also supported.

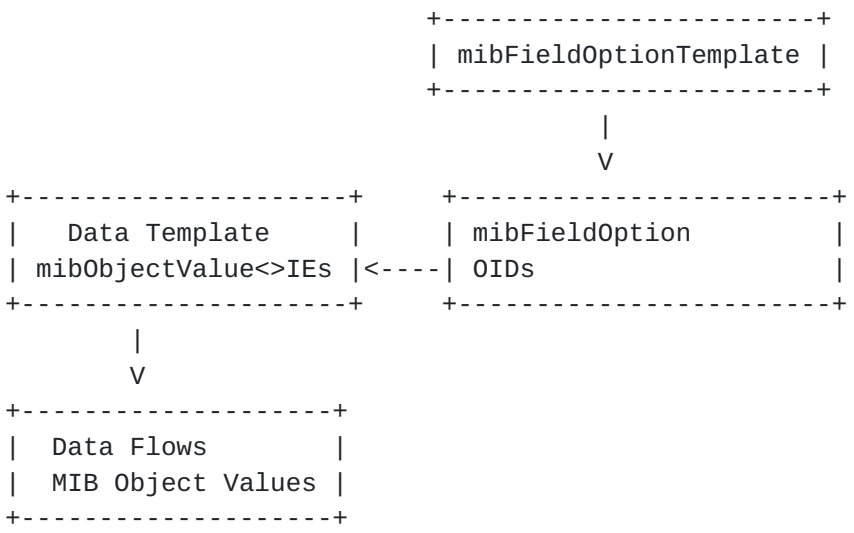

Figure 1: Architectural Overview

One common type defined in the SMIv2 are SEQUENCEs or conceptual tables. It is desirable that exporting a complete or partial SEQUENCE is simple and efficient. This is accomplished by having a alternative option export that can export only the OID of the sequence as a whole.

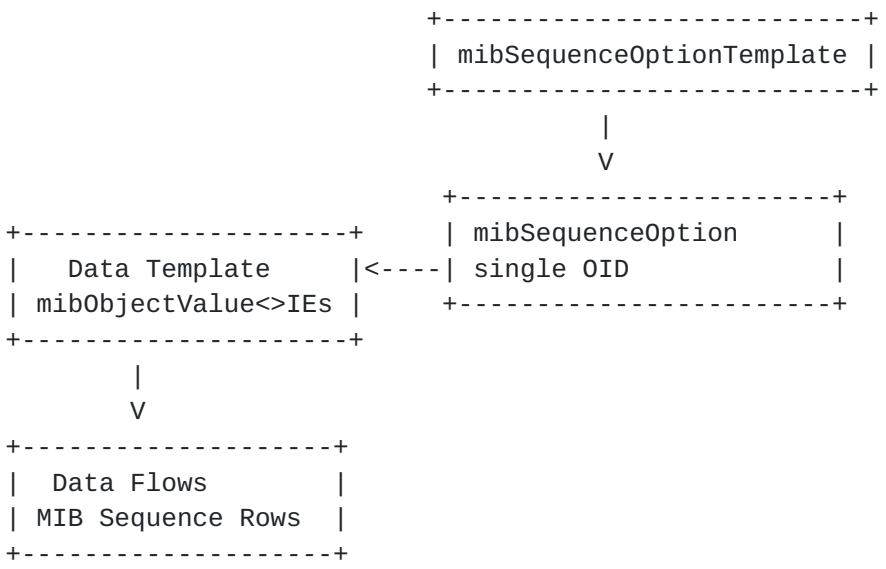

Figure 2: Architectural Overview Sequence

When individual MIB columner objects are 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

specification encompasses the different index types for the MIB Object 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)
- indexed by a string

A set of example use cases illustrates 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 OIDs of the objects that are used to index a MIB object. In such cases the Exporter can send the the values of the index OIDs 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:

- It requires implementing a mediation function between two data models, i.e., MIB objects and IPFIX Information Elements.
- Confirming the validity of simple mappings (e.g., ifIndex to ifName) requires either checking on a regular basis that the Exporter's network management system did not reload, or imposing ifIndex persistence across an Exporter's reload.
- 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 cannot be correlated. In the best case, assuming very tight integration of an IPFIX Collector with and 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 Data Records, in order to complement the Records with useful and already standardized information. More specifically, the case of an existing Template Record, which needs 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 Record containing the ingressInterface Information Element, augmented by interface counters [RFC2863], which are indexed by the respective ingressInterface values in the Data 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 the context of this specification. Indeed, newly created MIB objects (for example, the link utilization MIB variable), created with the Distributed Management Expression MIB [RFC2982] could nicely complement Data 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.

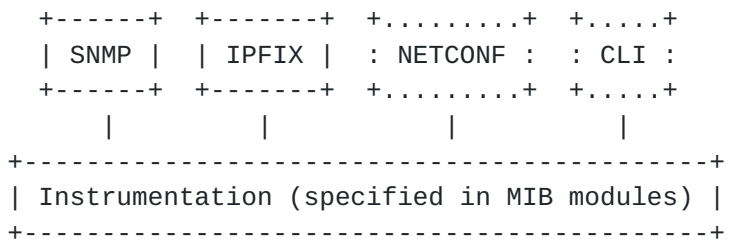

Figure 3: Architectural Model

The overall architectural model is depicted in Figure 3. 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 [RFC7011]. As in [RFC7011], 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 sequence 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 ("mibObjectIdentifier") which denotes that a MIB Object Identifier (MIB OID) is exported in the (Options) Data Record.

mibObjectValue<>

Refers to any and all of the mibObjectValue IEs generically. Any restriction or requirment in this document that refers to mibObjectValue<> applies to the following IEs: mibObjectValueInteger, mibObjectValueOctetString, mibObjectValueOID, mibObjectValueBITS, mibObjectValueCounter,

mibObjectValueGauge, mibObjectValueTime, mibObjectValueUnsigned, mibObjectValueTable andmibObjectValueSequence.

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 Object Value Information Element and the MIB Field Option Template

This document defines new mibObjectValue<> Information Elements (in Section 11). These are used to export MIB Objects as part of standard IPFIX Templates. The mibObjectValue<> Information Elements contains the actual data values retrieved from a MIB Object.

The issue that arises from exporting MIBs in IPFIX is that the standard IPFIX Template format ([RFC7011]) only provides the Collector with the length of the Information Element. The actual MIB OID would be unknown, so every MIB Field would appear identical and the contents of the MIB field would be incomprehensible data to a Collector.

For the values in the mibObjectValue<> fields to be understandable, more meta information about the mibObjectValue<> must be sent as part of the IPFIX export. The required minimum to undertand each field is the OID as defined in Section 5.1.2.

One approach to this problem would be to extend the IPFIX standard to allow Extended Field Specifiers so metadata about Fields can be included in Data Templates. This would however require a new version of the IPFIX standard which may not be backwards compatible. However, future versions of IPFIX MAY export the required MIB metadata as part of newer set versions.

This document defines a mibFieldOption Template to export the extra meta information required for a mibObjectValue<> field. This is a standard IPFIX Option Template Set that MUST include a minimum set of required fields (see Section 5.1.2) and MAY include extra fields to provide more meta information about one of the mibObjectValue<> fields.

5.1. MIB Field Option Template Format

For each mibObjectValue<> field that is defined in an IPFIX Data or Option Template, a mibFieldOption Data Record MUST be exported that provides the required minimum information to define the MIB object that is being exported (see Section 5.1.2). This mibFieldOption is

defined in a template referred to in this document as a mibFieldOption Template and has the format that is specified in Section 5.3.

A Template that uses a mibObjectValue<> Field MUST be exported in the same IPFIX message as the corresponding mibFieldOption Template and Data Records. Note that this places an implicit size constraint on the export.

5.1.1. Use of field order in the MIB Field Option template

The mibFieldOption export makes use of the informationElementIndex to specify which field in the template that the metadata relates to, which avoids any ordering constraints on the data template. The MIB field and any fields used to index it can be in any order in the export packet.

The informationElementIndex specifies which Field in the Template extra information is being provided for.

This is analogous to standard IPFIX Template Sets which also specify the order of the fields and provide their type and size.

If the template changes such that the order is different then the mibfieldOption data MUST be resent to reflect the new ordering. A new Template id MUST be used to reflect that the ordering has changed. Otherwise older mibFieldOption details may be used to refer to the incorrect field.

5.1.2. Minimum Required MIB Object Fields

To unambiguously export a mibObjectValue<> Field 3 fields are REQUIRED:

- (scope) templateId
- (scope) informationElementIndex
- mibObjectIdentifier (Section 11.1.2.1)

These are the minumum fields required in a mibFieldOptionTemplate Section 5.3.3.

While the following are optional, they are nevertheless RECOMMENDED in certain circumstances as they are per field.

- mibCaptureTimeSemantics (Section 11.1.2.3)

- mibIndexIndicator (Section 11.1.2.2)
- mibContextEngineID (Section 11.1.2.4)
- mibContextName (Section 11.1.2.5)

There are also fields that provide type information from a MIB Object definition that MAY be exported to a CP. Since these values are statically defined in the MIB they are not expected to change frequently. However the additional information about the MIB may help a CP that does not have access to the MIB.

- mibObjectName (Section 11.1.3.1)
- mibObjectDescription (Section 11.1.3.2)
- mibObjectSyntax (Section 11.1.3.3)
- mibModuleName (Section 11.1.3.4).

5.2. MIB Field Option Architecture

Four IPFIX Sets are used together to export a Flow using the mibObjectValue<> Field. These are:

1. A Data or Option Template Set which includes the mibObjectValue<> field.
2. A mibFieldOption Template Set
3. mibFieldOption Data Set
4. Data Set.

The Template Set or Option Template Set informs the Collector that a MIB Object Value of length n will be exported.

The mibFieldOption Template describes which metadata will be sent for each mibObjectValue<> being exported.

The mibFieldOption Data Set includes the metadata for each MIB object (ie the mibObjectIdentifier). The metadata about the mibObjectValue<>s only needs to be resent as per normal Template refreshes or resends.

The Data Set contains only the actual data extracted from the MIB.

Figure 4 shows the IPFIX Message structure for a MIB Field in a Template Set, while Figure 5 shows the IPFIX Message structure for a MIB Field in an Option Template Set.

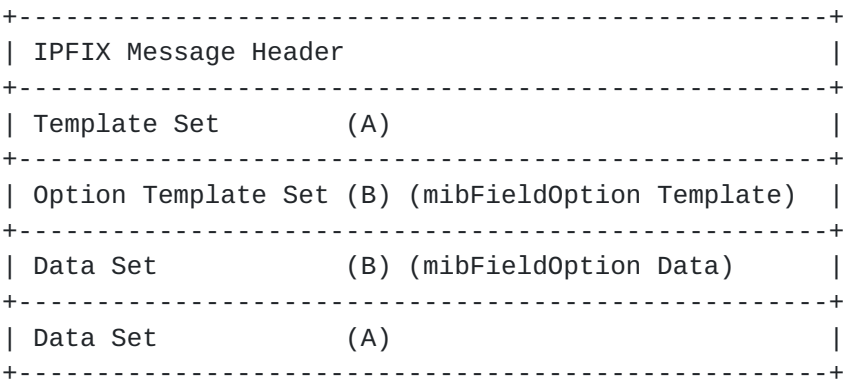

Figure 4: IPFIX Message structure for a MIB Field in a Template Set

The mibFieldOption Template can be used with Standard Option Templates as well.

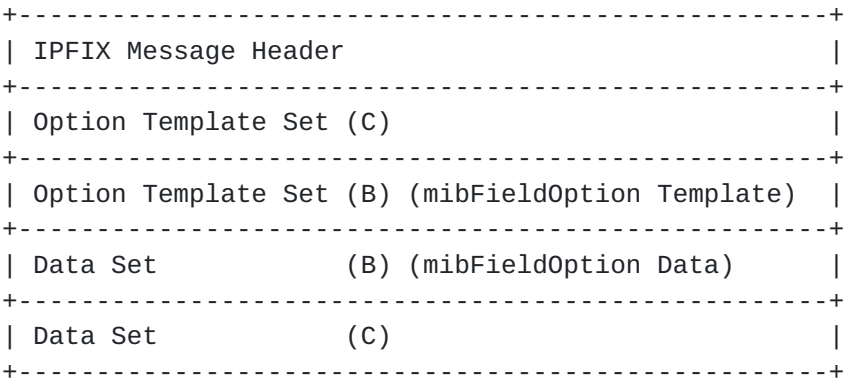

Figure 5: IPFIX Message structure for a MIB Field in an Option Template Set

More Data Sets that use the mibObjectValue<> can then be send in subsequent packets.

Figure 6 shows the relationships between the Records discussed above.

The mibFieldOption Template defines mibFieldOption Records. The mibFieldOption Data record annotate the Data Template with mibObjectValue<> metadata. Together the Data Template and mibFieldOption define the Data Records that will be exported.

The Data Records X have a dependency on the 2 Templates and the mibFieldOption Data Records.

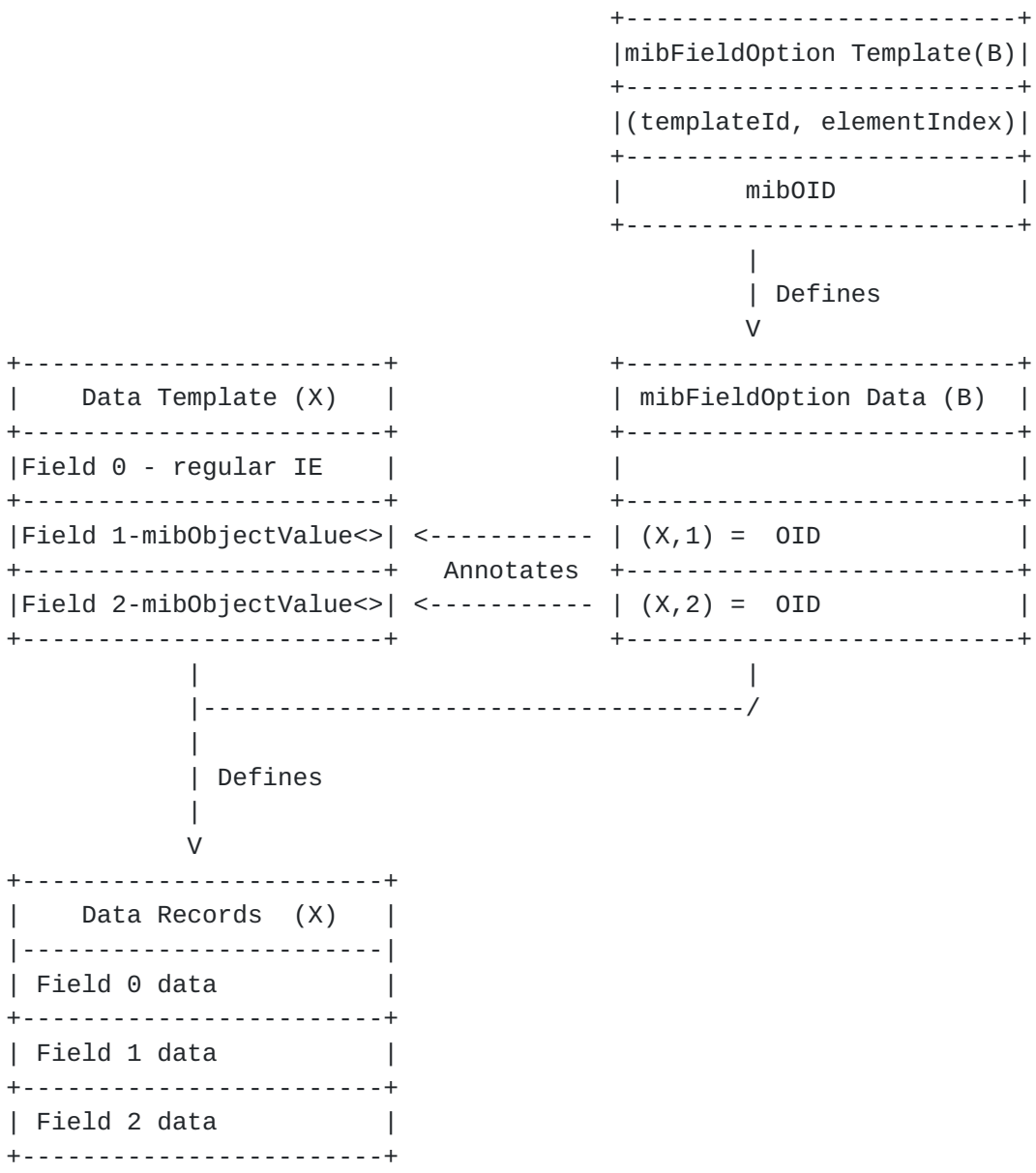

Figure 6: Relationships between Sets

5.3. MIB Field Option Template Formats

5.3.1. Data Template containing a MIBObject Field

The Template Record Format of a Template that uses the mibObjectValue<> field is identical to the standard IPFIX Format as defined in [RFC7011], so the mibObjectValue<> field is specified using standard IPFIX Field Specifiers as in [RFC7011].

The only extra requirement on a Template Record using mibObjectValue<> Fields is that it MUST export the required metadata

specified for EACH mibObjectValue<> Field (see Section 5.1.2). Multiple mibFieldOption MUST NOT be exported that refer to a single mibObjectValue<>.

There is a one to one mapping between each mibObjectValue<> field and a mibFieldOption.

A mibFieldOption Template and corresponding Data Record MUST be exported to provide the minimum required metadata.

Figure 7 shows an IPFIX Template Set using a mibObjectValue<> Field.

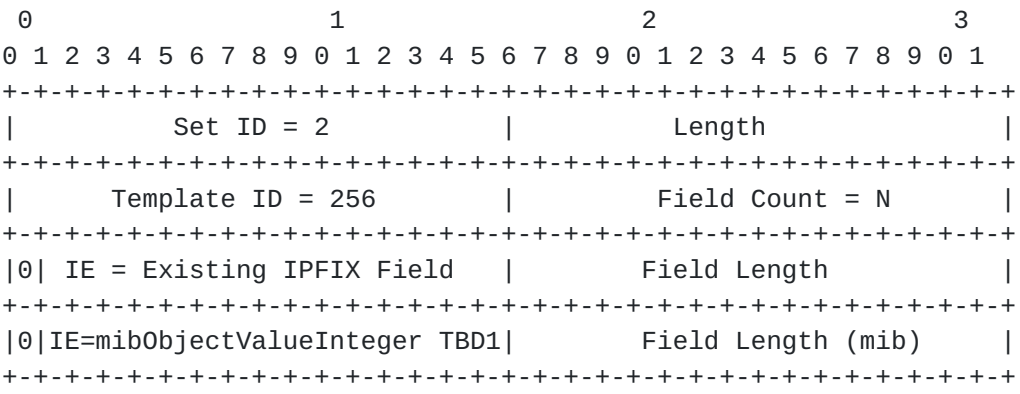

Figure 7: IPFIX Template Set using mibObjectValue<> Field

Where:

mibObjectValueInteger

One of the mibObjectValue<> IPFIX Information Elements that denotes that a MIB Object data (ie, the value of a MIB OID) will be exported in the (Options) Template Record. The specific type of the data in this case would be an Integer. Any other mibObjectValuee<> could be used to export a different MIB object type. When a mibObjectValue<> Information Element is used, the MIB Object Identifier ("mibObjectIdentifier") MUST be exported via a mibFieldOption or by other means. See Section 5.1.2.

Field Length (mib)

The length of the encoded MIB OID data in the corresponding Data Records, in octets. The definition is as [RFC7011]. Note that the Field Length can be expressed using reduced size encoding per [RFC7011]. Note that the Field Length may be encoded using variable-length encoding per [RFC7011].

5.3.2. Option Template containing a MIBObject Field

The Option Template Record Format of a Template that uses the mibObjectValue<> field is identical to the standard Format as defined in [RFC7011]. The mibObjectValue<> field is specified using standard Field Specifiers as in [RFC7011].

A mibObjectValue<> Field can be a Scope Field or a Non Scope Option Field.

The only extra requirement on a Option Template Record using mibObjectValue<> Fields is that it MUST export the required metadata specified in Section 5.1.2 for EACH mibObjectValue<> Field.

An IPFIX Option Template Record MUST export a mibFieldOption Template and Data Record to provide the minimum required metadata for each mibObjectValue<> Field.

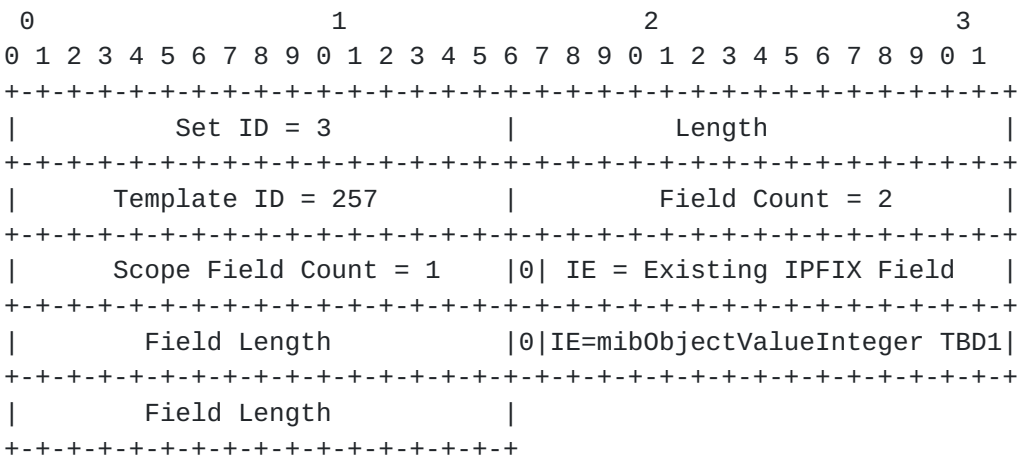

Figure 8: IPFIX Option Template Set using a Non Scope mibObjectValueInteger Field

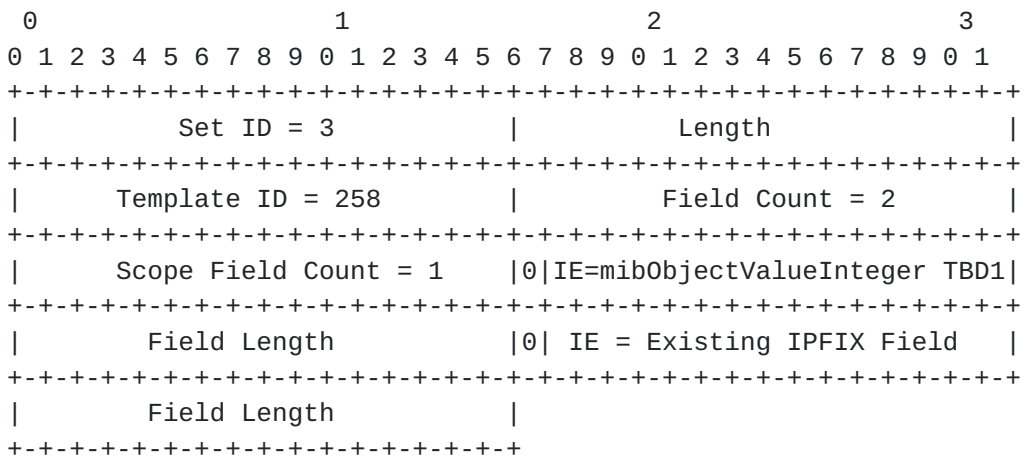

Figure 9: IPFIX Option Template Set using a Scope mibObjectValueInteger Field

5.3.3. mibFieldOption Template

The mibFieldOption Template is a Standard Option Template which defines the Fields that will be exported to provide enough metadata about a mibObjectValue<> so that the Collector can tie the data values in the mibObjectValue<> back to the definition of the MIB Object.

All mibFieldOption Templates MUST contain the following Fields:

- (scope) templateId
- (scope) informationElementIndex
- mibObjectIdentifier

A mibFieldOption Template MAY specify other Information Elements as part of the mibFieldOption template.

This document defines some common optional Information Elements which allow exporting more information about MIB Indexing, extra data from the MIB and the semantics of how the mibObjectValue<> was collected prior to export. See Section 11.1.

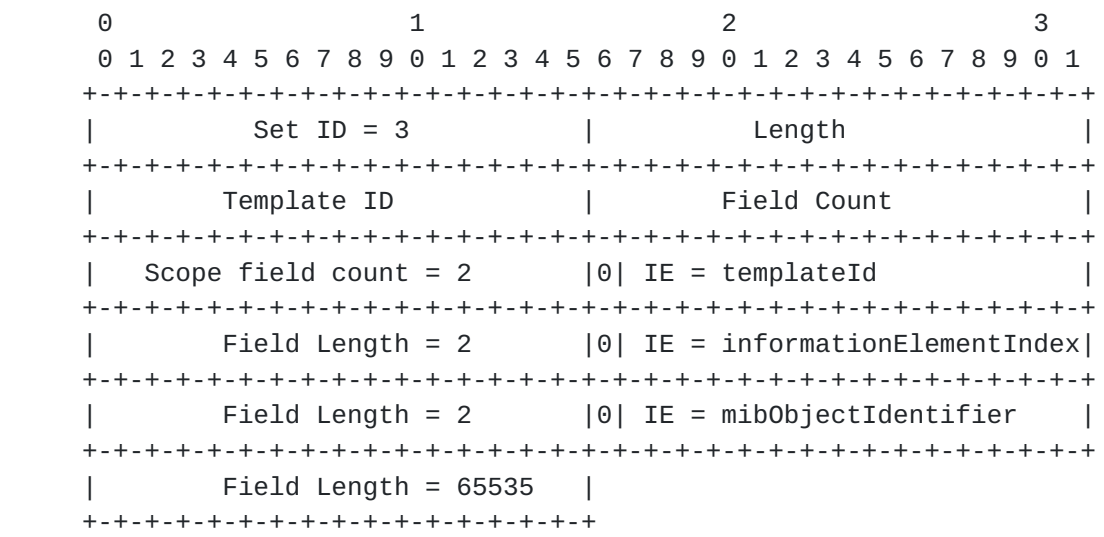

Figure 10: MIB Field Option Template Format - Required Fields

Where:

templateId

The first scope field is an IPFIX Information Element which denotes that a Template Identifier will be exported as part of the mibFieldOption Data Record. This Template Identifier paired with an index into that template, the "informationElementIndex" field, uniquely references one mibObjectValue<> Field being exported.

informationElementIndex

The second scope field is an IPFIX Information Element which denotes a zero based index into the fields defined by a Template. When paired with a "templateId" the Record will uniquely reference one mibObjectValue<> Field being exported.

mibObjectIdentifier

An IPFIX Information Element which denotes the a MIB Object Identifier for the mibObjectValue<> exported in the (Options) Template Record.

When the MIB Object Value Information Element is used, the MIB Object Identifier MUST be specified in the mibFieldOption Template Record or specified by other means.

The Object Identifier is encoded in the IPFIX data record in ASN.1/BER [BER] format.

Variable-length encoding SHOULD be used with the mibObjectIdentifier so that multiple different length MIB OIDs can be exported efficiently. This will also allow reuse of the mibFieldOption Template.

5.3.4. mibFieldOption Data Records

The mibFieldOption Data Records conform to the Template Specification in the mibFieldOption Template. There may be multiple mibFieldOption Records exported.

The Collecting Process MUST store all received mibFieldOption Data information for the duration of each Transport Session, because the Collecting Process will need to refer to the extra meta information to decode a mibObjectValue<> Field fully.

Figure 11 shows the format of the exported mib Field Option detailing the metadata that will be exported to match the Template in section (see Figure 10).

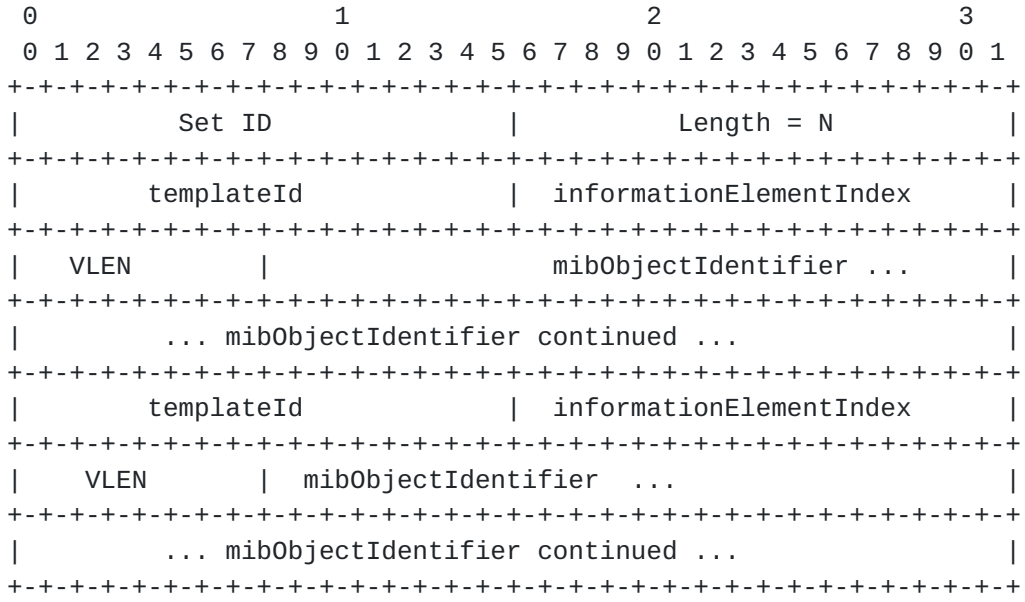

Figure 11: Format of mibFieldOption Data Record

5.3.5. mibFieldOption Template with Indexing

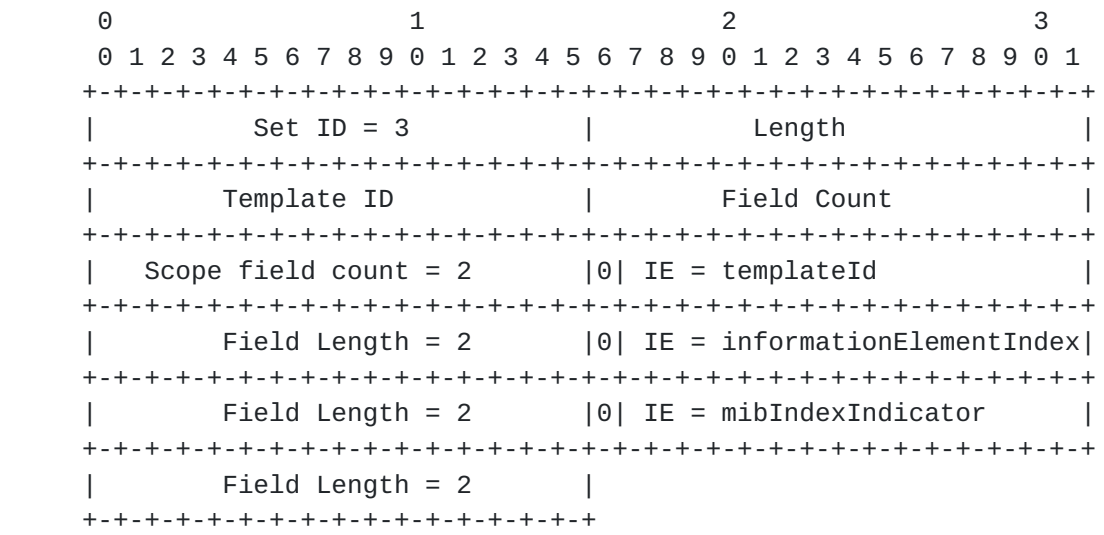

Figure 12: mibFieldOption Format for a indexed MIB Object

Where:

mibIndexIndicator

The MIB Index List IPFIX Information Element which marks which fields in the record will act as Index values for the exported MIB Object.

The index data for a mibObjectValue<> will be other fields contained in the same Data Record. The mibIndexIndicator marks the Fields whose value(s) should be added to the mibObjectIdentifier to completely index the value.

Elements used to Index MIB Objects MUST be exported in the same order as they are specified in the INDEX field of the SEQUENCE they belong to.

The Information Elements referred to by mibIndexIndicator could be any Information Element(s) that can be appended onto a the mibObjectIdentifier to index it fully.

See '7.7. Mapping of the INDEX clause'[RFC2578] for which MIB Objects will be suitable for INDEXing a Sequence and how they are appended to the fully index an OID.

If there are no other suitable Fields to index a mibObjectValue<>, then one or more mibSubIdentifier field can be included in the (Options) Template Record and refered to by the mibIndexIndicator. This allows the implementer some

flexiblility to export a MIB values with a simple IE that will always be an option.

5.3.6. mibFieldOption Template with Semantics Fields

A mibFieldOption Template MAY specify that extra Information Elements will be exported to record how the mibObjectValue<> was collected

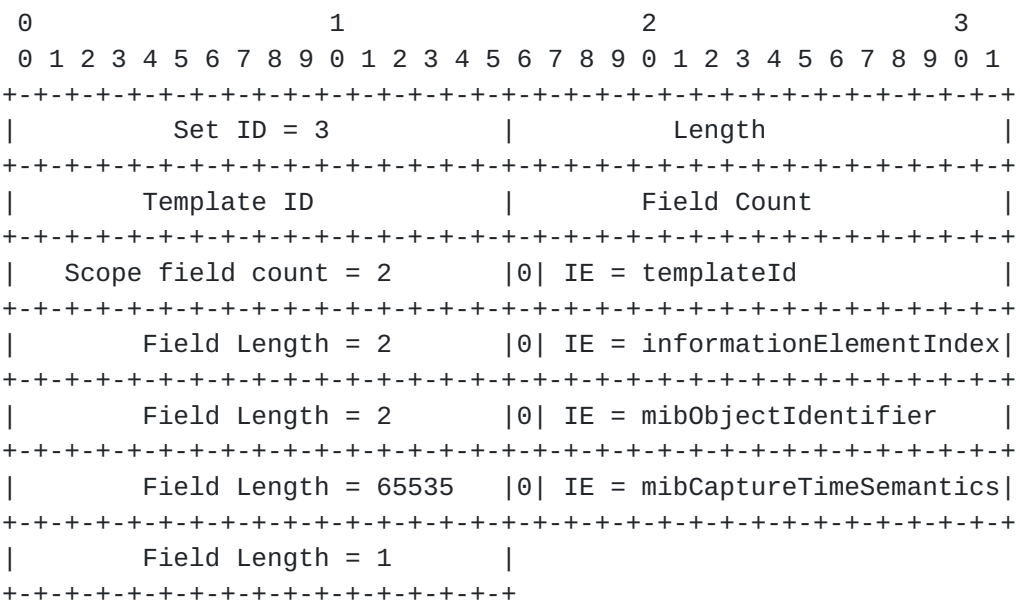

Figure 13: mibFieldOption Format for a non-indexed Field with Semantic Data

Where:

mibCaptureTimeSemantics

The MIB Capture Time Semantics IPFIX Information Element, as defined in Section 11.1.2.3.

It is RECOMMENDED to include this field when exporting mibObjectValue<>s that specify counters or statistics. In particular for situations with long lived Flows.

5.3.7. mibFieldOption Template with extra MIB Object Details

The MIB OID exported within the mibObjectIdentifier IPFIX Information Element provides a reference to a MIB that will fully describe the MIB Variable being Exported.

However an Exported Process MAY decide to include some extra MIB Fields to more fully describe the mibObjectValue<>.

This can be helpful if the Collecting Process may not have access to the MIB. It also allows the IPFIX Field Types to be extended with any MIB Variable already defined purely through IPFIX.

The exporting process can either include the extra Object Details fields as part of the mibFieldOption Template or export a separate Option Template and Data that maps MIB OIDs in mibObjectIdentifier Fields to the Object details.

If a single or few fields are being exported then including extra type data in the mibFieldOption export will be more efficient.

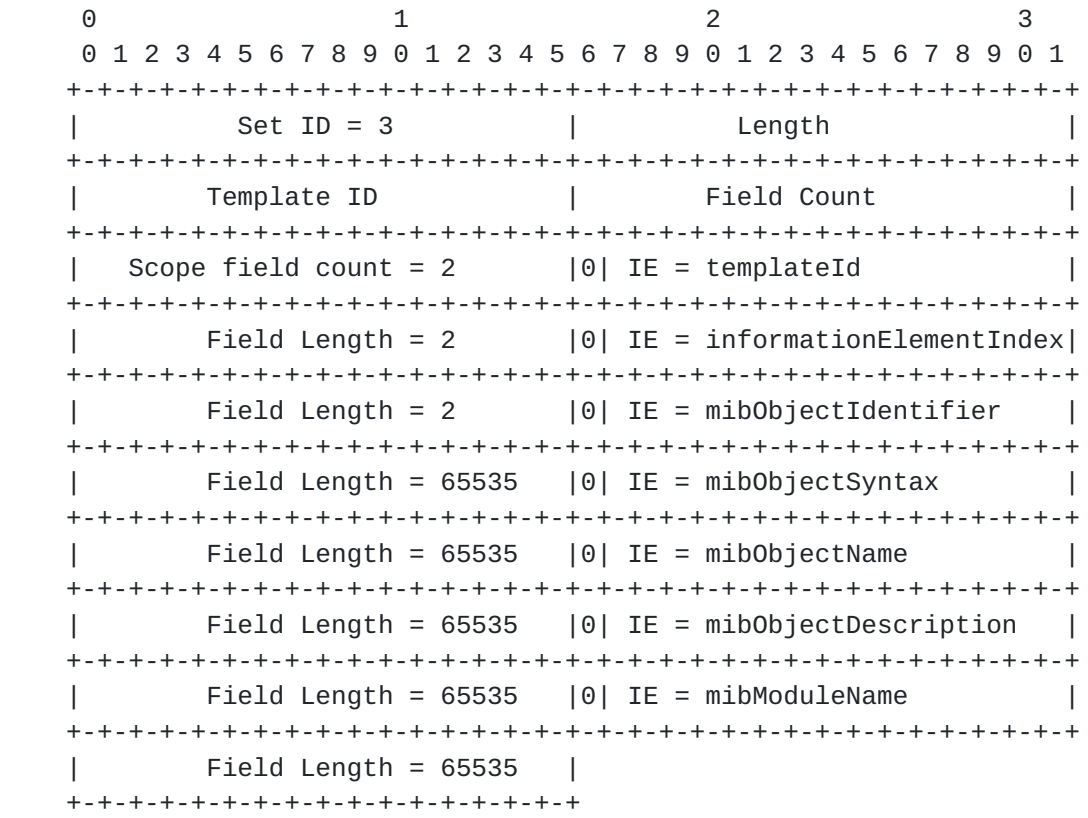

Figure 14: mibFieldOption Format for a non-indexed Field with Object Details

Where:

mibOjectSyntax

The MIB SYNTAX clause string for a mibObjectIdentifier, which contains the SYNTAX string as defined for the MIB Object referenced by the MIB OID. This will either be the name of one of the primitive "base types" from [RFC2578] or a Textual Convention name.

The SYNTAX clause may also include subtyping or specific enumerations for known values or flags.

mibObjectName

The textual name a mibObjectIdentifier Object.

mibObjectDescription

The textual description for a mibObjectIdentifier.

mibModuleName

The textual name of the MIB which defines a MIB OID Object.

Or the MIB details can be exported as an Standard Option Export as shown in Figure 15

This may be more efficent as the bulk of this data is text based and SHOULD be exported once to the CP if there are many MIB objects being exported. This prevents this large textual data being included for every use of a mibObjectValue<> field.

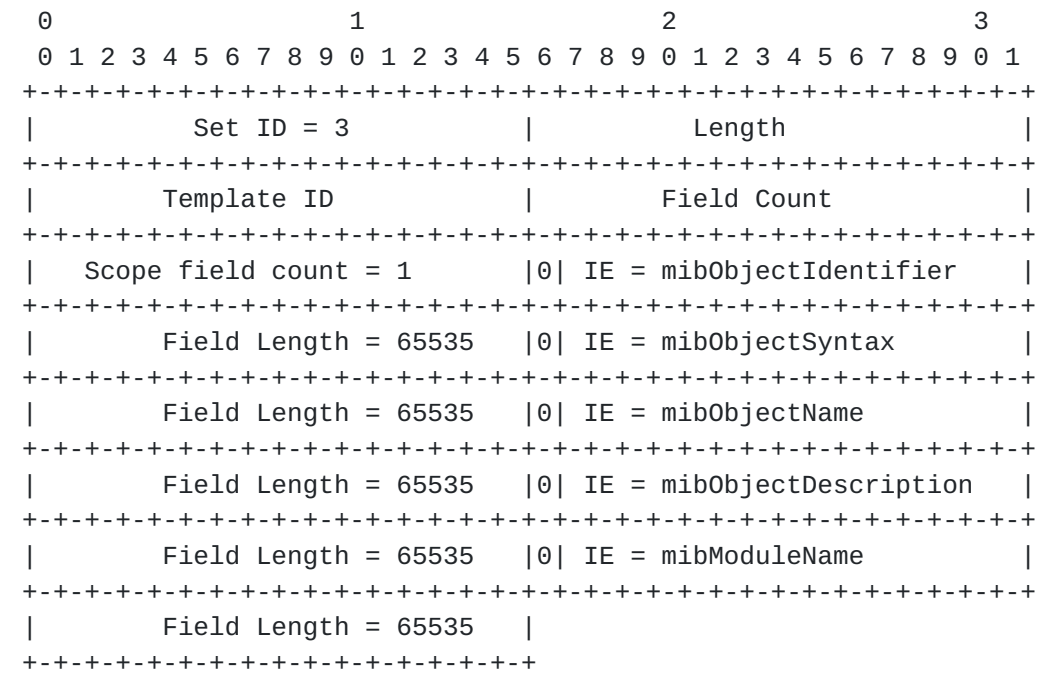

Figure 15: Alternative Option Export mibObjectIdentifier to Details

5.4. 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 MIB context MUST be identified by including the mibContextEngineID (Section 11.1.2.4) and mibContextName (Section 11.1.2.5) fields in the mibFieldOption Template and associated mibFieldOption data records.

This context data must be included for each field that is not in the default context.

TODO - example with contexts

See Section 11.

5.5. Template Management

Templates are managed as per [RFC7011] with the additional constraint that the mibFieldOption Template and Data Records MUST be exported in the same IPFIX Message as any (Option) Template Record that uses a mibObjectValue<>.

If a Template using mibObjectValue<>s is resent for any reason the Records it depends on MUST be sent as well.

If a Template is replaced with a new (Option) Template Record then a new mibFieldOption Data Record MUST be sent with the replacement referencing the new Template ID.

An Exporting Process SHOULD reuse mibFieldOption Template IDs IFF the Templates are identical. Each (Option) Template Record MUST still be accompanied by a copy of the mibFieldOption Template.

5.6. IPFIX and MIB Data Model

The RFC 2578 species that the SYNTAX clause for a MIB Object defines the abstract data structure of an Object and must contain:

"The data structure must be one of the following: a base type, the BITS construct, or a textual convention. (SEQUENCE OF and SEQUENCE are also possible for conceptual tables, see section 7.1.12)." [RFC2578] section-7.1

For each of these the options this draft specifies exactly which mibObjectValue<> to use.

If a MIB Object to be exported is a textual convention the definition of the textual convention must be consulted and the SYNTAX clause used to determine the correct base type. This may recurse if the textual convention is defined in terms of another textual convention but this should end at a base type.

If the SYNTAX clause contains a Textual Convention or Subtyping the mibOjectSyntax IE SHOULD be used to export this detail to the CP.

The Options for the SYNTAX are then mapped as follows:

RFC2578 Section No	SYNTAX	mibObjectValue<>
7.1.1	Integer32 and INTEGER	mibObjectValueInteger
7.1.2	OCTET STRING	mibObjectValueOctetString
7.1.3	OBJECT IDENTIFIER	mibObjectValueOID
7.1.4	The BITS construct	mibObjectValueBITS
7.1.5	IpAddress	mibObjectValueIPAddress
7.1.6	Counter32	mibObjectValueCounter
7.1.7	Gauge32	mibObjectValueGauge
7.1.8	TimeTicks	mibObjectValueTime
7.1.9	Opaque	mibObjectValueOctetString
7.1.10	Counter64	mibObjectValueCounter
7.1.11	Unsigned32	mibObjectValueUnsigned
7.1.12	Conceptual Tables	mibObjectValueTable or mibObjectValueSequence

Table 1: SMIv2 SYNTAX -> mibObjectValue types

Values are encoded as per the standard IPFIX encoding of Abstract Data Types. The only new encoding references in this document is that Object Identifiers (OID) will be encoded as per ASN.1/BER [BER] in an octetArray

5.7. Exporting Sequences

There are several approaches for an IPFIX exporting process to use MIB SEQUENCES also known as conceptual tables.

A columnar object may be used purely use as a data type. In this case no indexing or relation to a sequence provided by the exporting process.

As part of an option export of the contained and complete conceptual table. A SEQUENCE OF

In a mixed option/data IPFIX/MIB template where the mib value can be indexed by other fields in the record

Using structured data to export the complete "SEQUENCE OF" or individual rows of "SEQUENCE"s As part of a mibObjectValueSequence structured data export

This draft defines two forms of Indexing that can be used for SEQUENCE MIB Objects

FIELD based indexing is used by having every mibObjectValue<> export a mibIndexIndicator to flag the Index fields required. This allows complex indexing or mixing of existing IPFIX IE with MIB Fields. It also allows multiple MIB SEQUENCE values to be exported with complete indexing in 1 IPFIX Template.

SEQUENCE based Indexing is used by having a single Option Export that corresponds to the SEQUENCE or conceptual table. Each SEQUENCE of the MIB Object corresponds to a single Flow Record. The Index Fields defined in the INDEX clause of the MIB Object MUST all be present the same order as the Scope fields. This allows a simple table export of a MIB SEQUENCE without any extra indexing fields. There is a 1-to-1 mapping between the SEQUENCE and the option export so only the OID for the SEQUENCE must be exported.

Using structured data for Sequence export allows complete SEQUENCE 'conceptual rows' or SEQUENCE OF 'conceptual tables' to be exported completely as a single field in a record. The structured data fields mibObjectValueTable and mibObjectValueSequence always use SEQUENCE based Indexing

5.7.1. Exporting Sequences - Complete

Exporting a complete MIB Sequence is done with an IPFIX Option Template Set. All the scope fields for the option record are used as the index fields. The mibIndexIndicator is not required in the mibFieldOption record.

When mibSequenceOption indexing of a sequence is used all scope fields MUST be the INDEX Objects in the same order as defined in the MIB SEQUENCE being exported.

Any extra INDEX fields that are part of the SEQUENCE MUST be included in the scope fields of the template if they occur in the INDEX clause before any field in the row. If the INDEX fields are not the initial

MIB Objects in the sequence or are not in the same order in the INDEX and the SEQUENCE then mibSequenceOption MUST not be used and the complete information for the table MUST be exported using individual mibFieldOption records and mibIndexIndicator fields.

The mibObjectValue<> fields exported using the mibSequenceOption Sequence export MUST all be members of the same SEQUENCE Object.

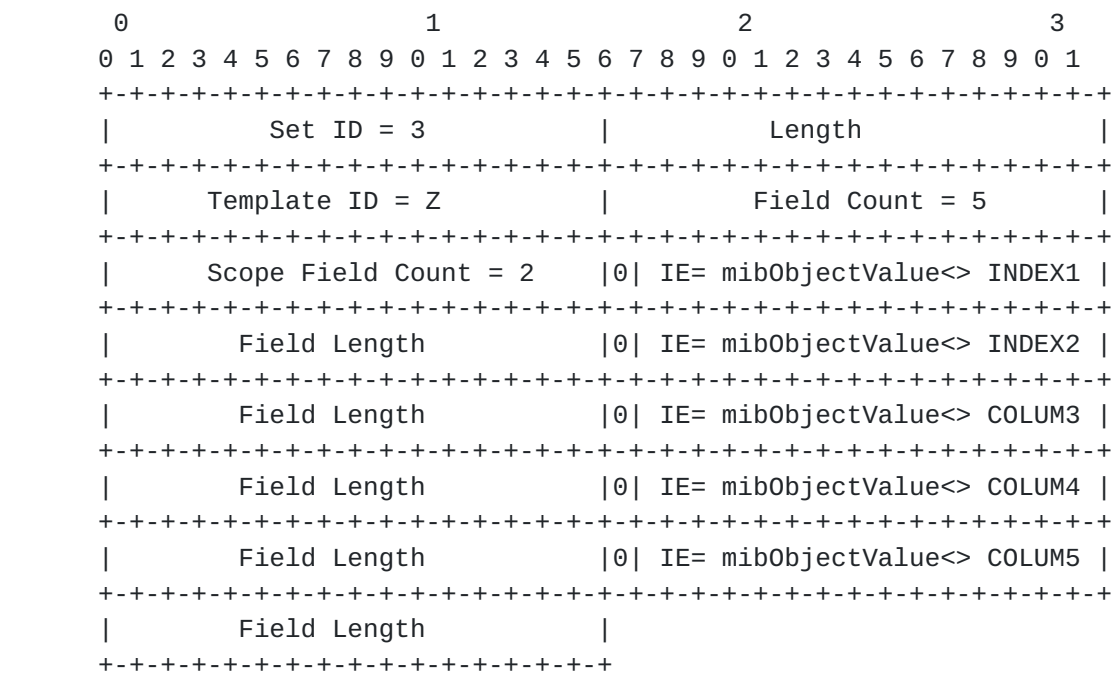

Figure 16: IPFIX Option Template for a complete SEQUENCE with 5 columns and 2 INDEX fields

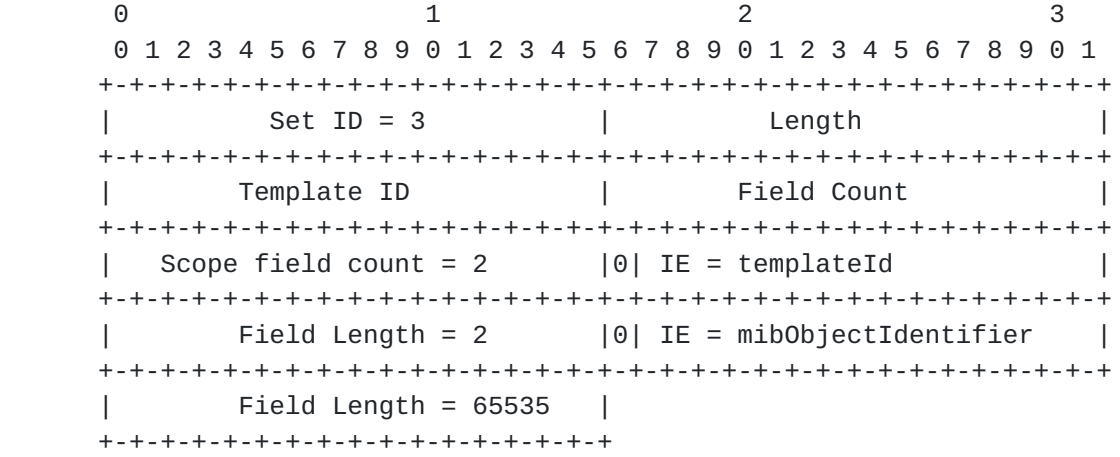

Figure 17: mibSequenceOption Format for a SEQUENCE Object

Where:

templateId

The templateId for the MIB Option that will be exported.

mibObjectIdentifier

The MIB OID for the SEQUENCE that is being exported with this mibSequenceOption.

5.7.2. Exporting Sequences - Structured Data

TODO. see mibObjectValueTable and mibObjectValueSequence.

5.7.3. Exporting Sequences - Explicit

The other option for indexing a sequence is explicit indexing. In this case there may be non index fields in the scope of the option export or there may be multiple mib objects being exported with different indexes. In this case each mib field requires the mibIndexIndicator with the bits set for the fields that are used to index that individual Object. The index fields MUST be in the 'correct' order as defined in the SEQUENCE that the columnar object is a member of.

See section Section 5.3.5. TODO merge these sections together.

6. Example Use Cases

6.1. 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 2: Established TCP Connections

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

1. flowStartSeconds from [RFC7012], 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 18 shows the exported Template Set detailing the Template Record for exporting the number of established TCP connections (see Section 6.1).

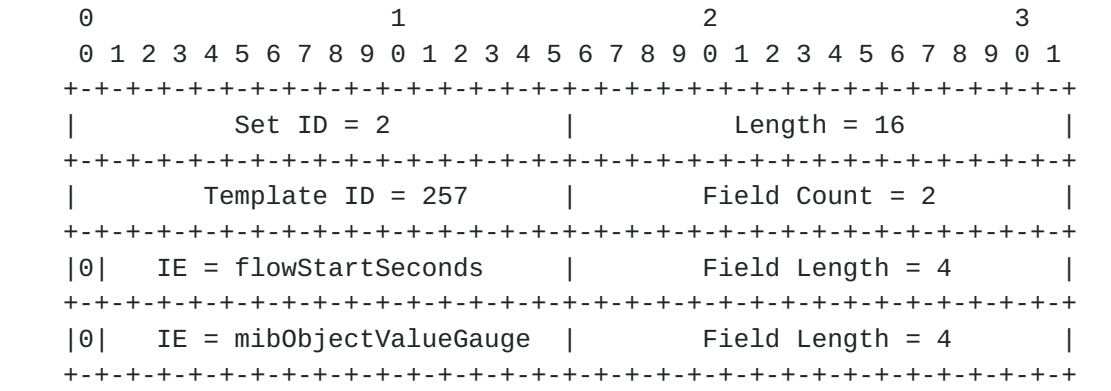

Figure 18: Example of tcpCurrEstab Template Set

Figure 19 shows the exported mib Field Option detailing the metadata that will be exported about the mibObjectValueGauge Field in Template 257 in Template Record (see Section 6.1).

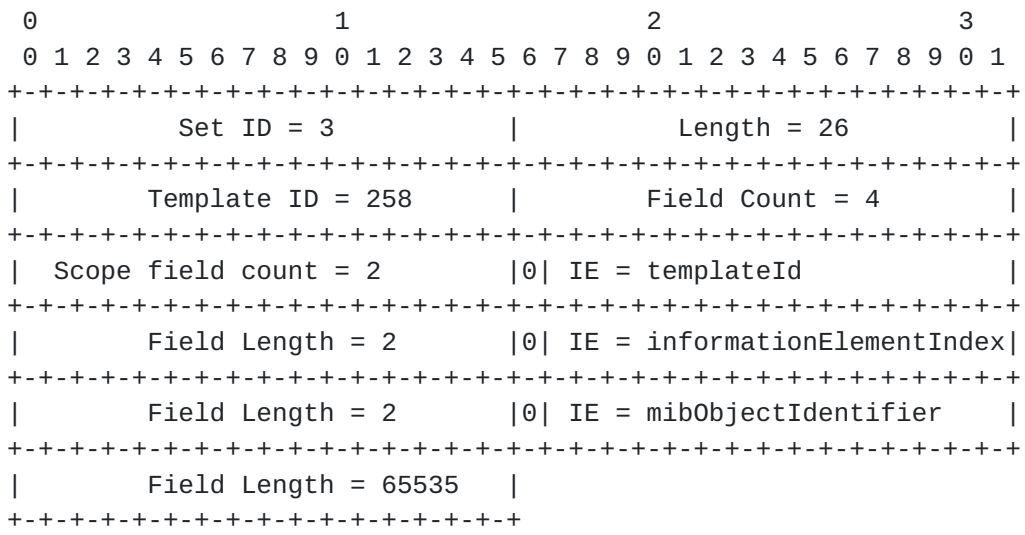

Figure 19: Example of tcpCurrEstab mibFieldOption Template Set

Figure 20 shows the exported mib Field Option detailing the metadata that will be exported about the mibObjectValueGauge Field in Template 257 in Template Record (see Section 6.1).

The OID for the MIB Object tcpCurrEstab from [RFC4022], Object ID "1.3.6.1.2.1.6.9", will be encoded in ASN.1/BER [BER] as '06072b060102010609' in the data record. This will take 9 bytes.

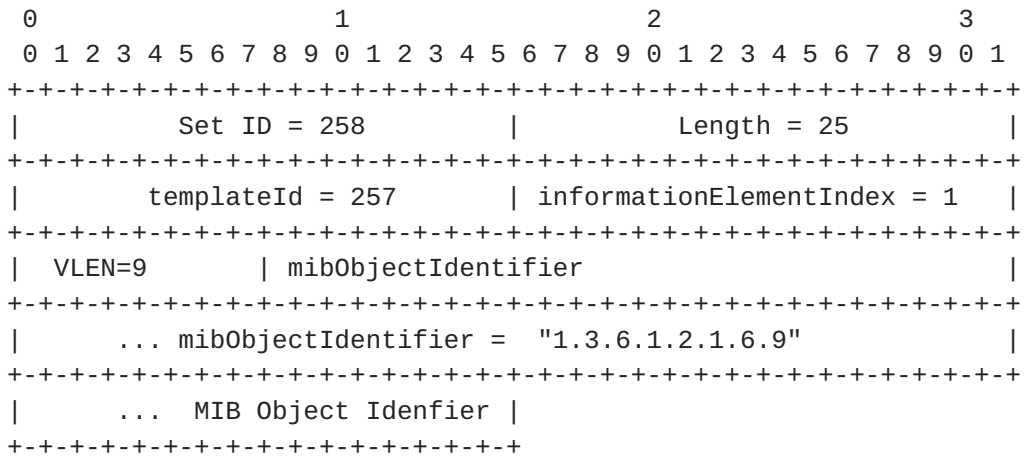

Figure 20: Example of tcpCurrEstab mibFieldOption Data Set

Figure 21 shows the start of the Data Set for exporting the number of established TCP connections (see Section 6.1).

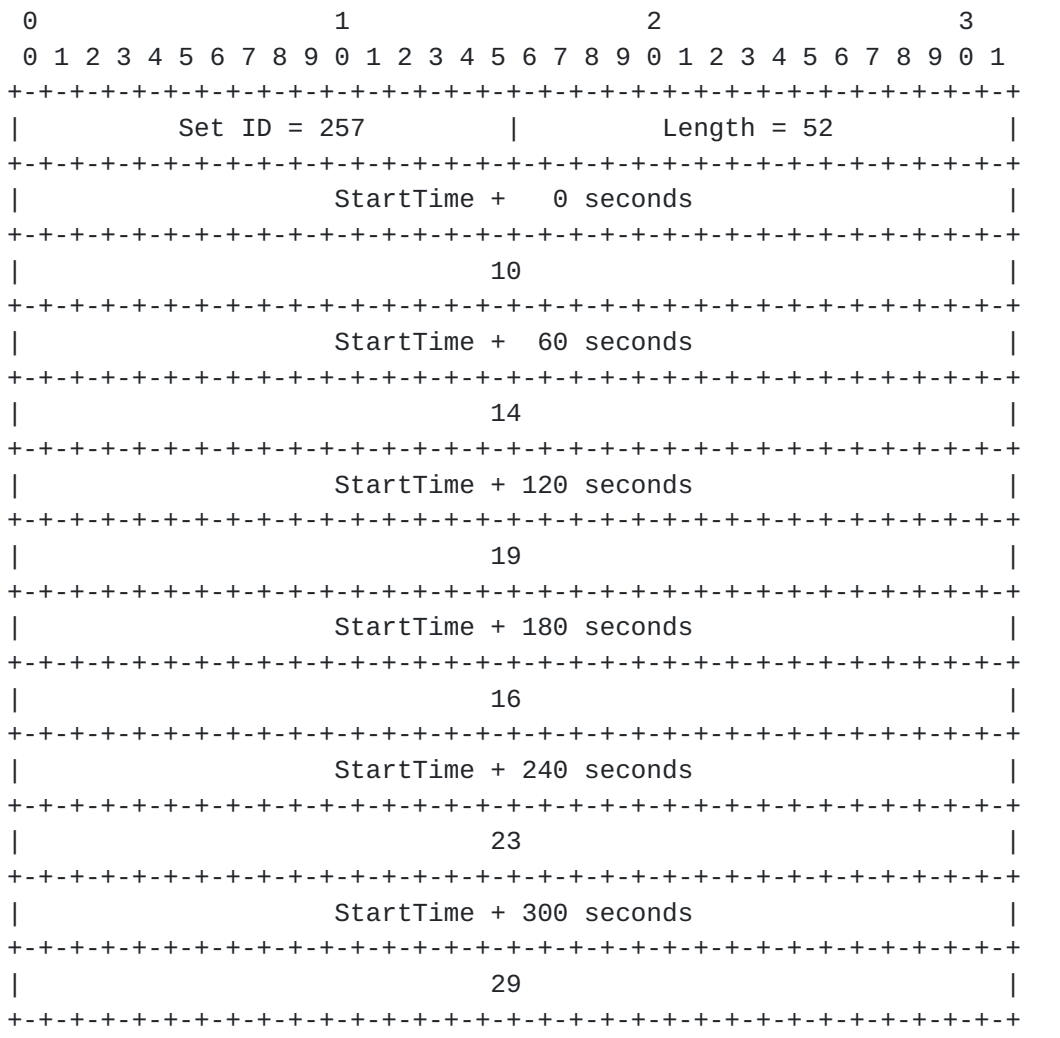

Figure 21: Example of tcpCurrEstab Data Set

6.2. Enterprise Specific MIB Object: Detailing CPU Load History

For the sake of demonstrating a 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 cpmCPUTotal1minRev 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 3: CPU Usage Data

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

1. flowStartSeconds from [RFC7012], Information Element 150: The absolute timestamp of the first packet of this Flow.

2. a mibObjectValueGauge for cpmCPUTotal1minRev, the overall CPU busy percentage in the last one-minute period

Figure 22 shows the exported Template Set detailing the Template Record for exporting CPU Load (see Section 6.2).

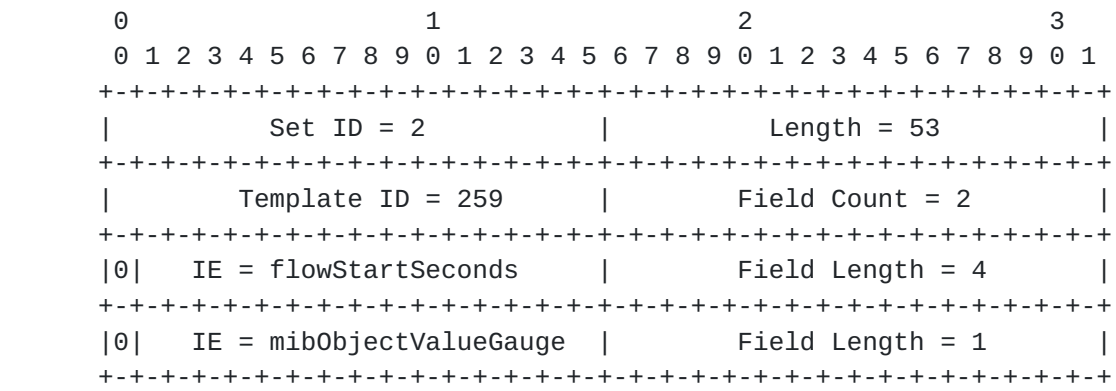

Figure 22: Example of CPU Load Template Set

Figure 23 shows the exported Template Set detailing the mibfieldOption Template Record for exporting CPU Load (see Section 6.2). Note this identical to the mibFieldOption template given in Figure 19 so the same template could have been reused.

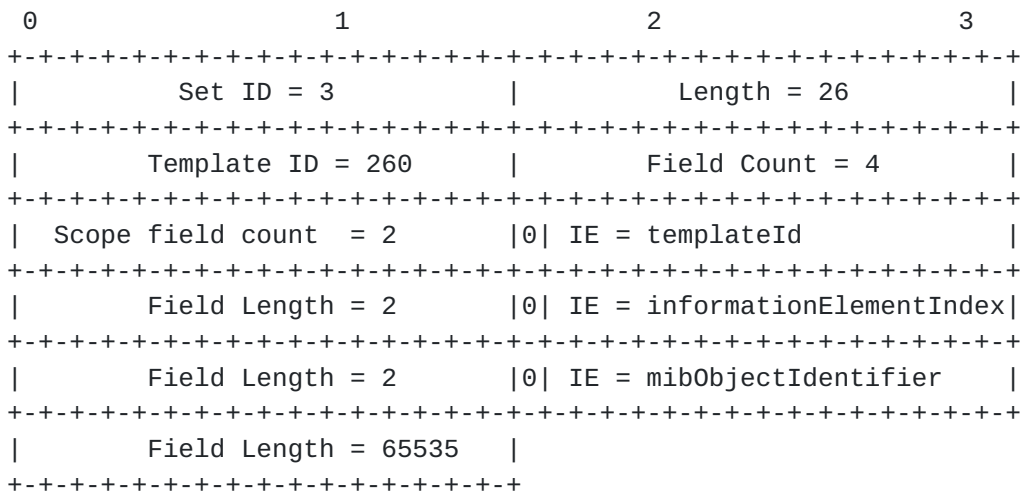

Figure 23: Example of CPU Load mibFieldOption Template Set

Figure 24 shows the exported mib Field Option detailing the metadata that will be exported about the mibObjectValueGauge Field in Template 259 in Template Record (see Section 6.2).

The OID for the cpmCPUTotal1minRev has been encoded using ASN.1/BER to '060d2b0601040109096d0101010107' at 15 bytes long.

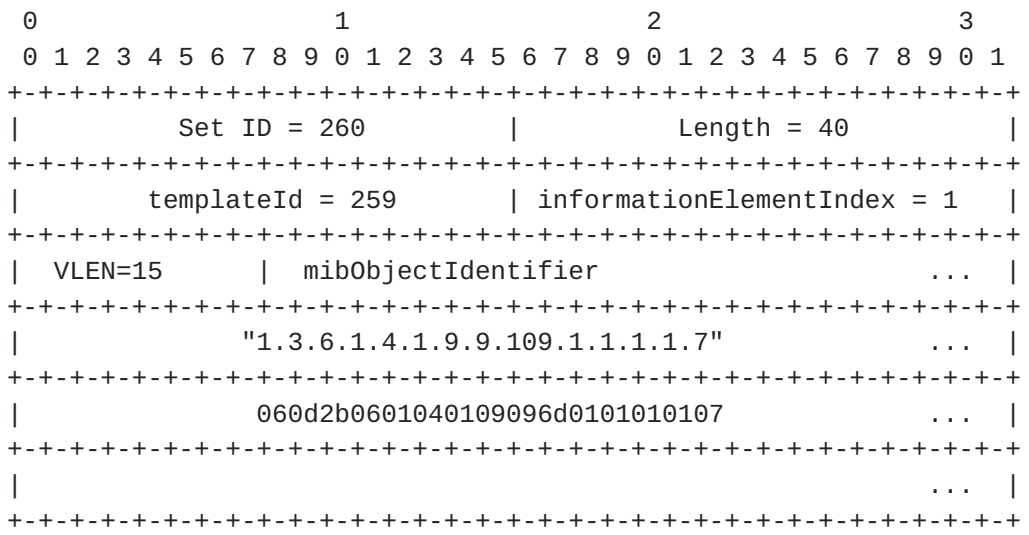

Figure 24: Example of CPULoad mibFieldOption Data Set

Note that although cpmCPUTotal1minRev is 32 bits long, reduced size encoding ([RFC7011]) has been used to encoded it within a single octet.

This example stresses that, even though the OID cpmCPUTotal1minRev is enterprise-specific, the E bit for the mibObjectValueGauge and

mibObjectIdentifier is set to "0" since the "mibObjectValueGauge" and "mibObjectIdentifier" Information Element is not enterprise-specific. That this data is from an Enterprise MIB is included in the OID that includes an Enterprise ID.

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

6.3. Exporting A Complete Sequence Table: The OSPF Neighbor Sequence

Many conceptual tables are already defined in standard and proprietary MIBs. These can be exported with a minimum of overhead by using the mibSequenceOption. This allows the exporter to unambiguously define an export of an entie sequence as an Option Export. The use of mibSequenceOption means that each individual columnar object does not need to have it's OID exported to the collector

The ospfNbrTable defined in TODO OSPF MIB is a Sequence Of ospfNbrEntry, which has the OID "1.3.6.1.2.1.14.10.1". Each option data record will therefore correspond to an ospfNbrEntry.

The following fields will be exported:

Object	mibObjectValue	Length in bytes
ospfNbrIpAddr	mibObjectValueIPAddress	4
ospfNbrAddressLessIndex	mibObjectValueInteger	4
ospfNbrRtrId	mibObjectValueIPAddress	4
ospfNbrOptions	mibObjectValueInteger	1
ospfNbrPriority	mibObjectValueInteger	1
ospfNbrState	mibObjectValueInteger	1
ospfNbrEvents	mibObjectValueCounter	4
ospfNbrLSRetransQLen	mibObjectValueGauge	4
ospfNBMANbrStatus	mibObjectValueInteger	1
ospfNbmaNbrPermanence	mibObjectValueInteger	1
ospfNbrHelloSuppressed	mibObjectValueInteger	1
ospfNbrRestartHelperStatus	mibObjectValueInteger	1
ospfNbrRestartHelperAge	mibObjectValueUnsigned	4
ospfNbrRestartHelperExitReason	mibObjectValueInteger	1

Table 4: OSPF Neighbor Entry Objects

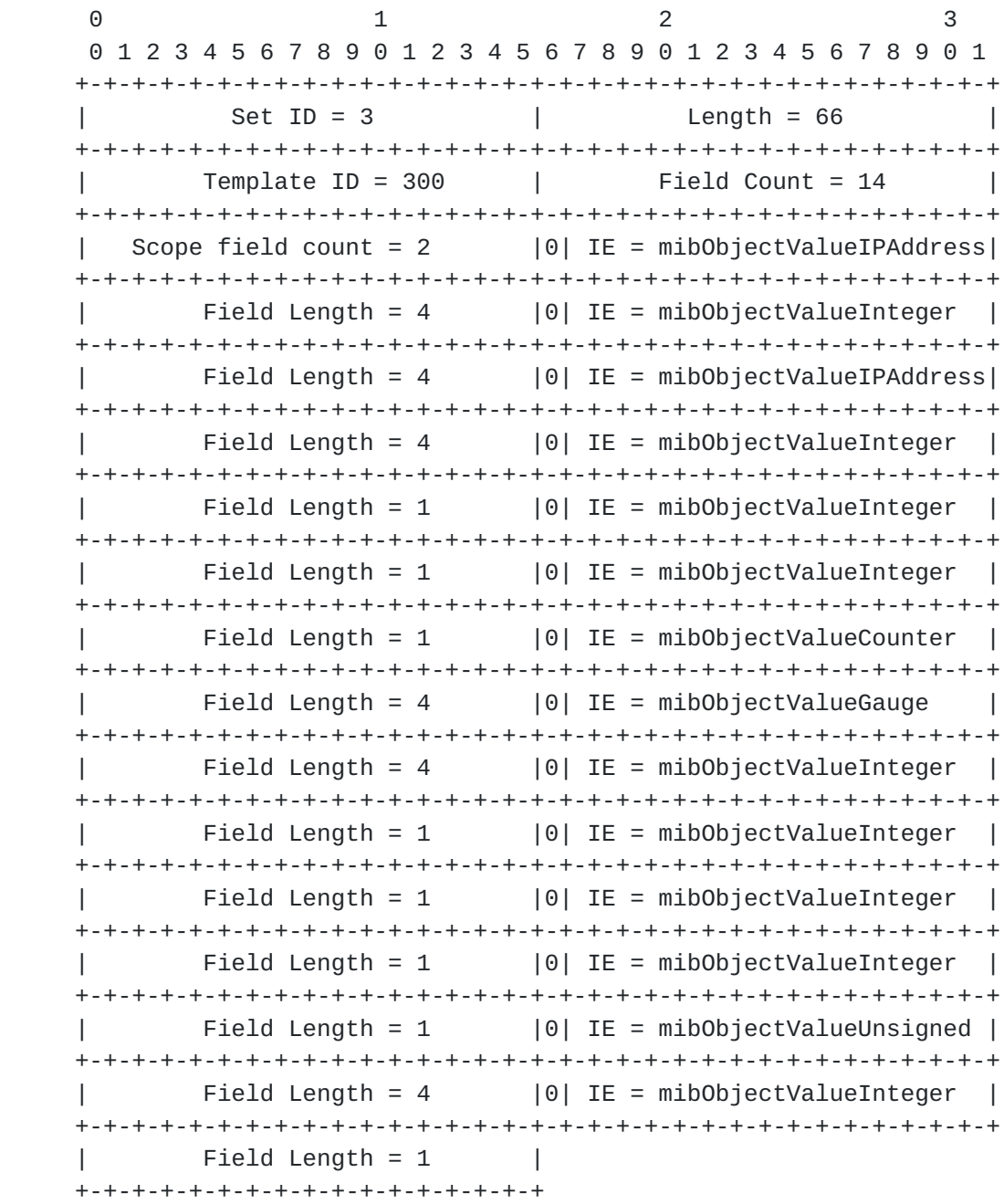

Figure 25: Example of ospfNbarEntry Template Set

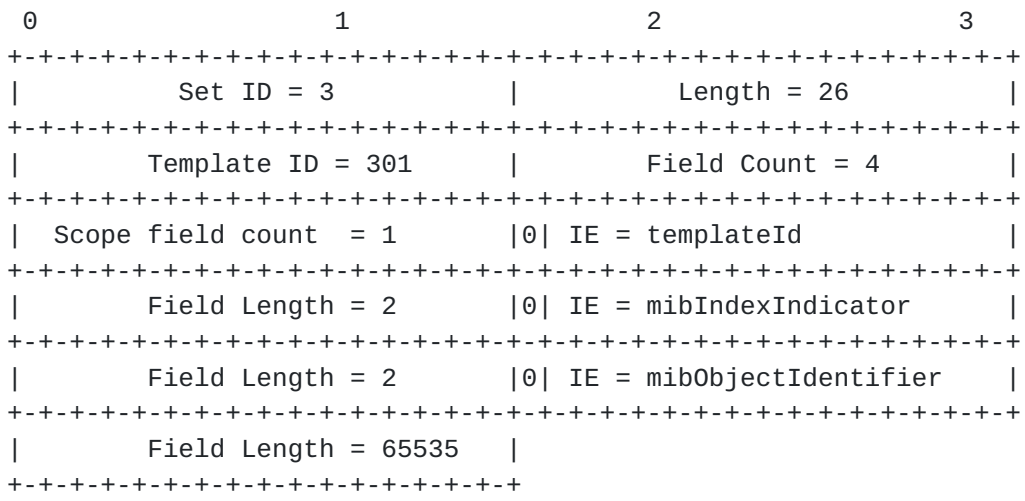

Figure 26: Example of ospfNbarEntry Sequence mibSequenceOption Template Set

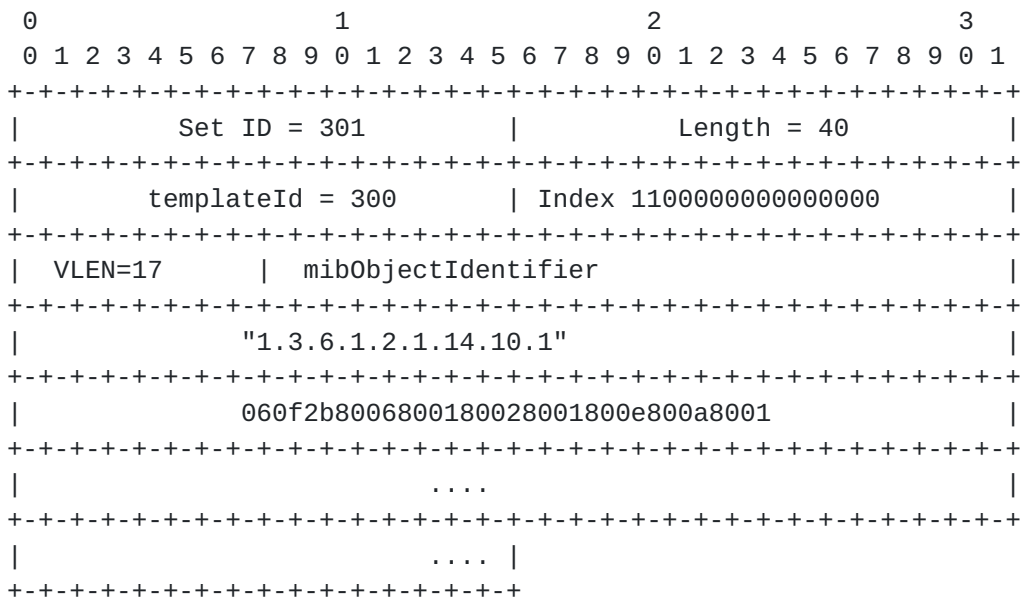

Figure 27: Example of ospfNbarEntry mibSequenceOption Data Set

TODO - Generate some data for this example

6.4. Exporting Selected Objects Sequence : The ipIfStatsInForwDatagrams

It may be that the full set of columnar objects that are supported by a SEQUENCE are not required to be exported. In this case the Objects specified by the INDEX clause of the SEQUENCE MUST be exported in the correct order and refered to by the mibIndexListIE included in the mibFieldOption Template.

This example shows the export of ipIfStatsInForwDatagrams from the IP-MIB [RFC4293]. ipIfStatsInForwDatagrams is a columnar object that is part of the conceptual table ipIfStatsTable SEQUENCE. This is comprised of ipIfStatsEntry conceptual rows.

The ipIfStatsTable SEQUENCE is indexed by the ipIfStatsIPVersion and ipIfStatsIfIndex, therefore to export ipIfStatsInForwDatagrams clearly they should be included.

Since textual conventions are being used the SYNTAX is also being exported as part of the mibFieldOption

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)

Figure 28 shows the exported Options Template Set.

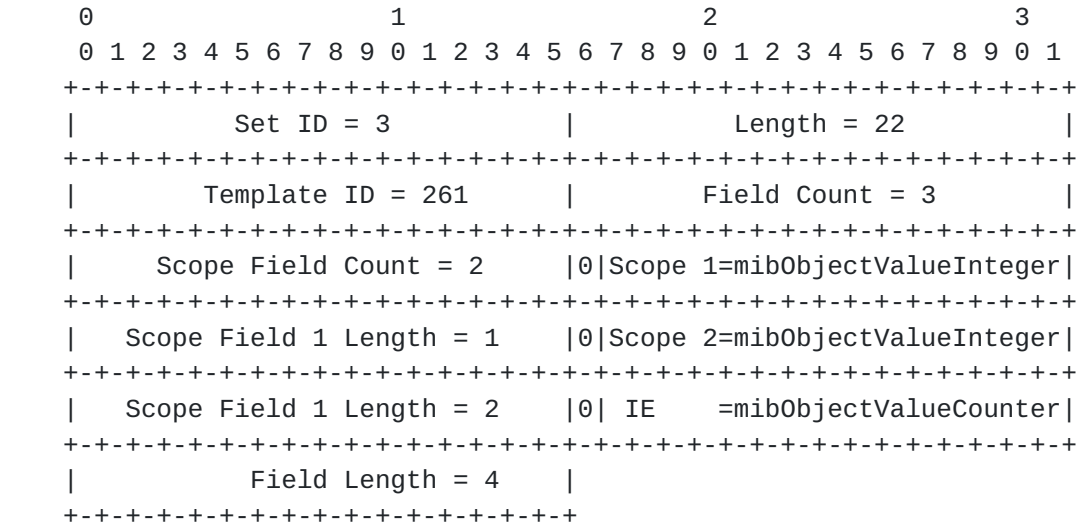

Figure 28: Example of an Options Template for an Indexed MIB Object with two indices.

Figure 29 shows the exported mibFieldOptions Template used to export the required mibObjectValue<> metadata. This example of the mibFieldOption Template includes the mibIndexList to indicate that some of the other fields in the data records are indexes.

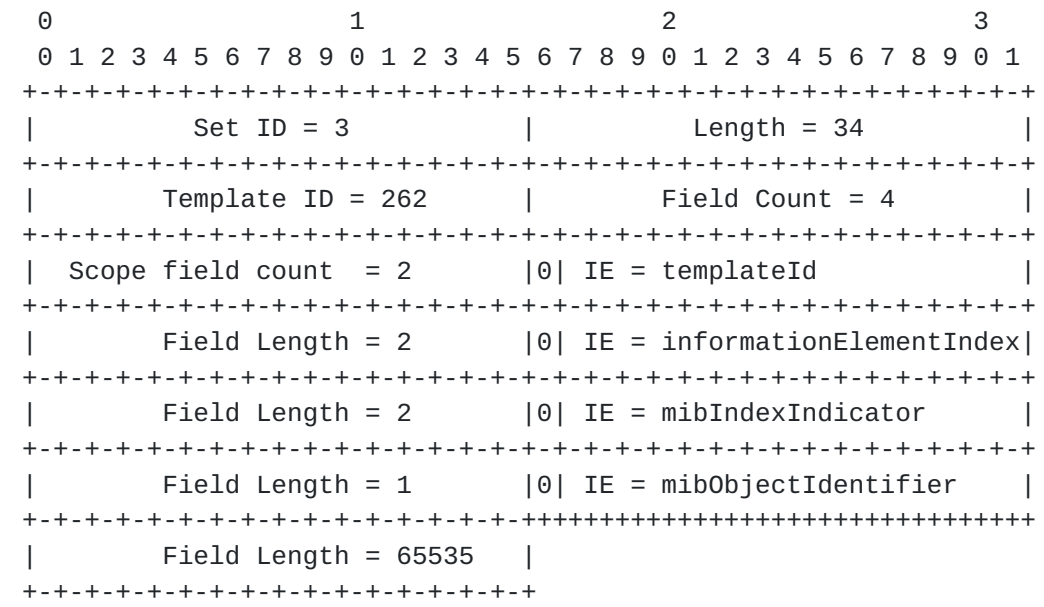

Figure 29: Example of an MIB Field Options Template for an Indexed MIB Object with two indices.

Figure 30 shows the exported mibFieldOption Data used to export the required mibObjectValue<> metadata. Note that the first 2 Data Records have their mibIndexIndicator length set to an empty length of 0. The third mibIndexIndicator has the value '11000000' to show that the first 2 fields in the record are the Index's for this columnar object.

Figure 30: Example of an MIB Field Options Data Set for an Indexed MIB Object with two indices.

Figure 31 shows the Data records that export the values of the 3 mibObjectValue<>.

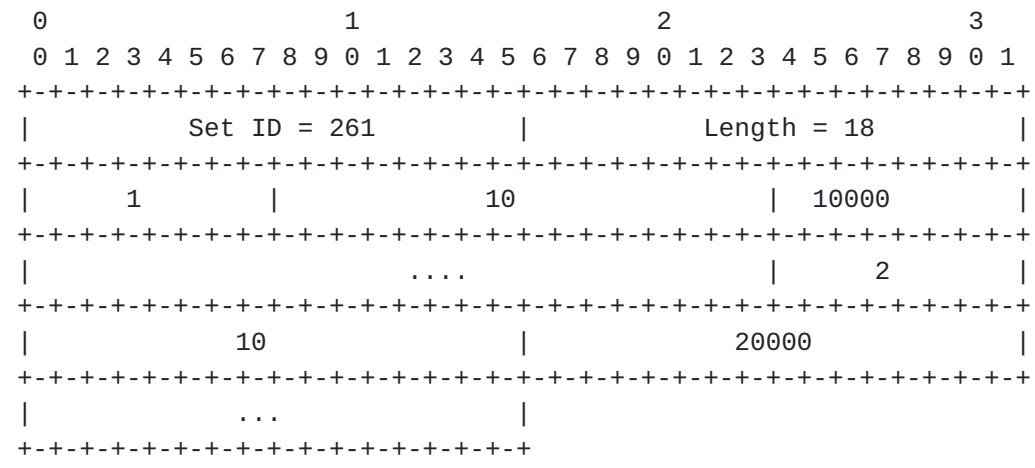

Figure 31: Example of an MIB Data Set for an Indexed MIB Object with two indices.

6.5. 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 be 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 5: 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"), this is part of the ifEntry SEQUENCE that has as it's index 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".

This relationship between the ifOutQLen field and the index field is exported using mibIndexIndicator in the mibFieldOption. The value of "00010000" flags the index fields concisely.

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 32 shows the exported Template Set detailing the Template for exporting a PSAMP Report with Interface Output Queue Length (ifOutQLen). Figure 33 and Figure 34 show the mibFieldOption Template and Data Record.

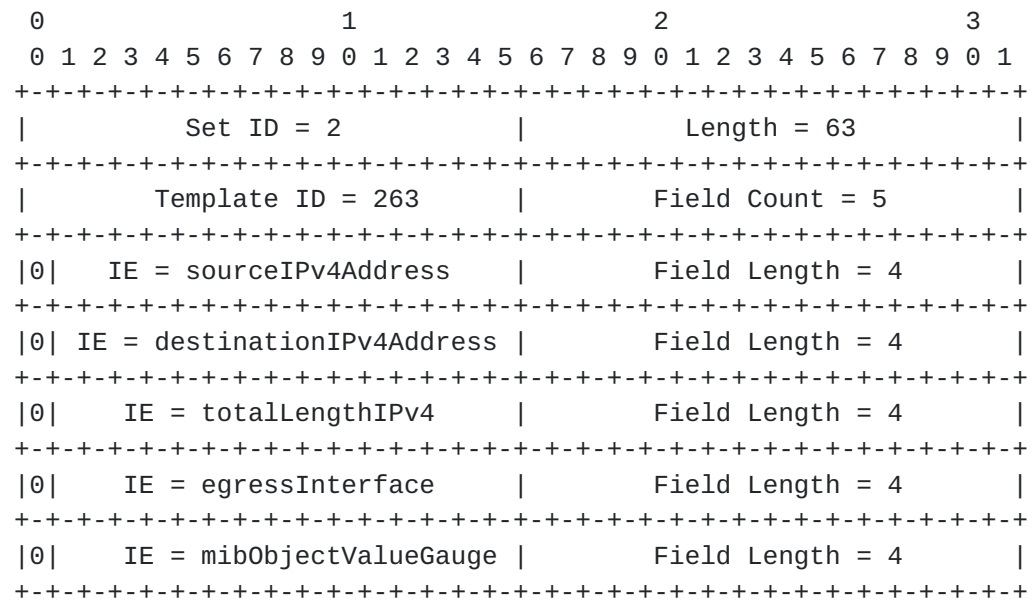

Figure 32: Example of Template for a PSAMP Report with ifOutQLen indexed by egressInterface

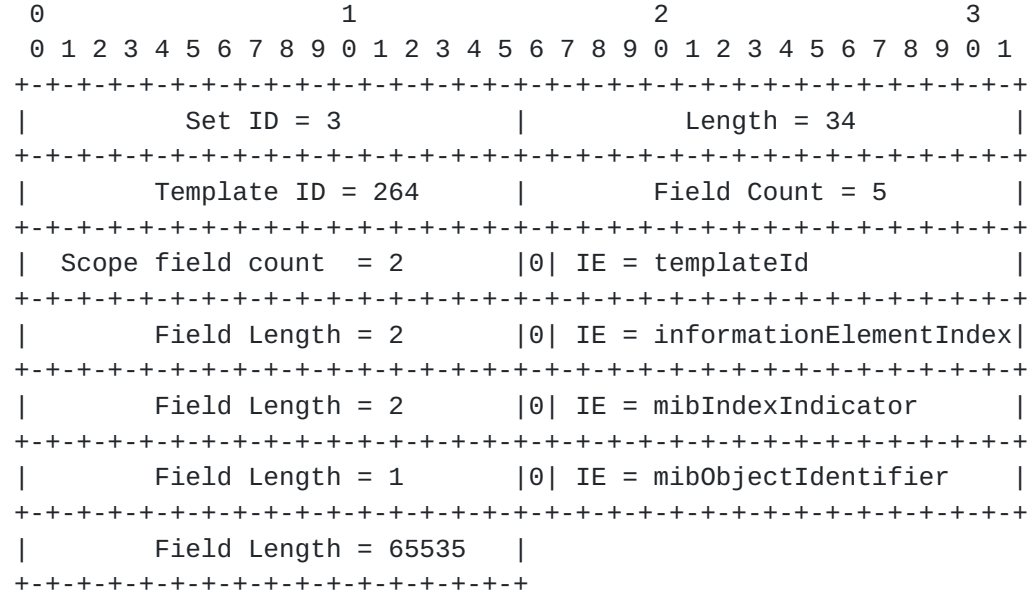

Figure 33: Example of mibFieldOption Template for a PSAMP Report with ifOutQLen indexed by egressInterface

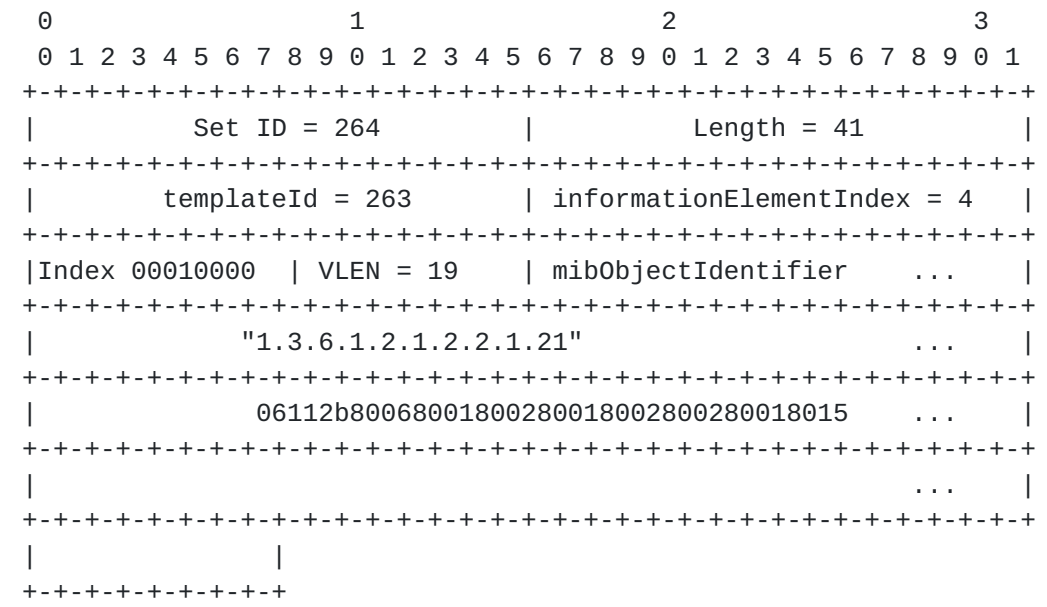

Figure 34: Example of mibFieldOption Data Record for a PSAMP Report with ifOutQLen indexed by egressInterface

The corresponding IPFIX Data Record is shown in Figure 35. 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.

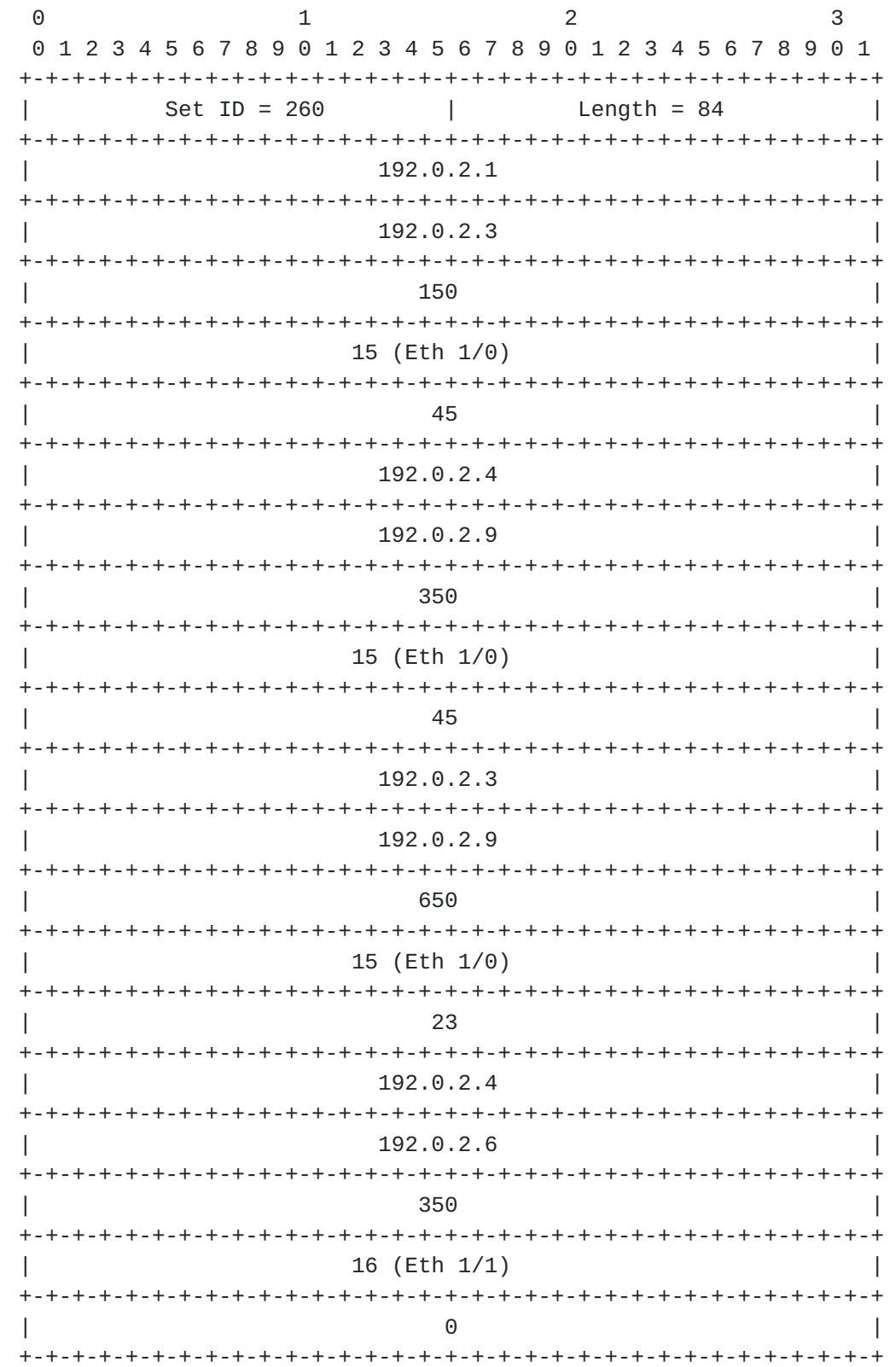

Figure 35: Example of PSAMP Packet Report with ifOutQLen indexed by egressInterface

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

Following on the example from the previous section (see Section 6.5), if the Template Record for the example Data Record does not contain the egressInterface, the ifOutQLen can be indexed by the ifIndex interface index as detailed in the IF-MIB [RFC2863] by including as many mibSubIdentifier as are required.

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

1. sourceIPv4Address

2. destinationIPv4Address

3. totalLengthIPv4

4. ifOutQLen indexed by: the mibSubIdentifier

5. mibSubIdentifier

Note the only significant difference between this and the previous example is that by using the mibSubIdentifier the exporter is exposing less information about the type of the Index.

Figure 36 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.

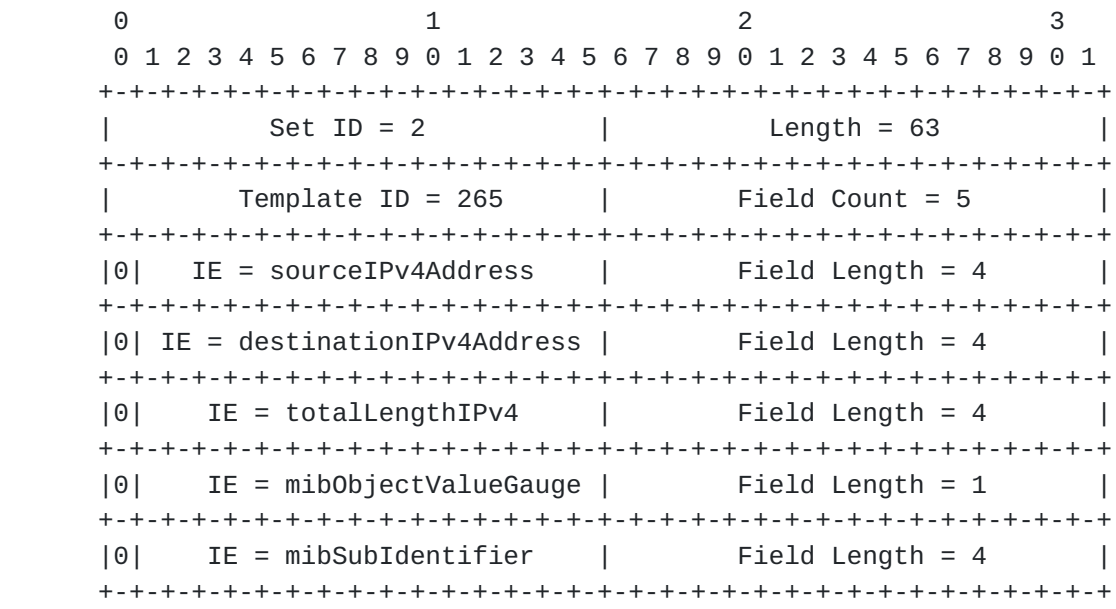

Figure 36: Example of Template for a PSAMP Report with ifOutQLen indexed by a string index

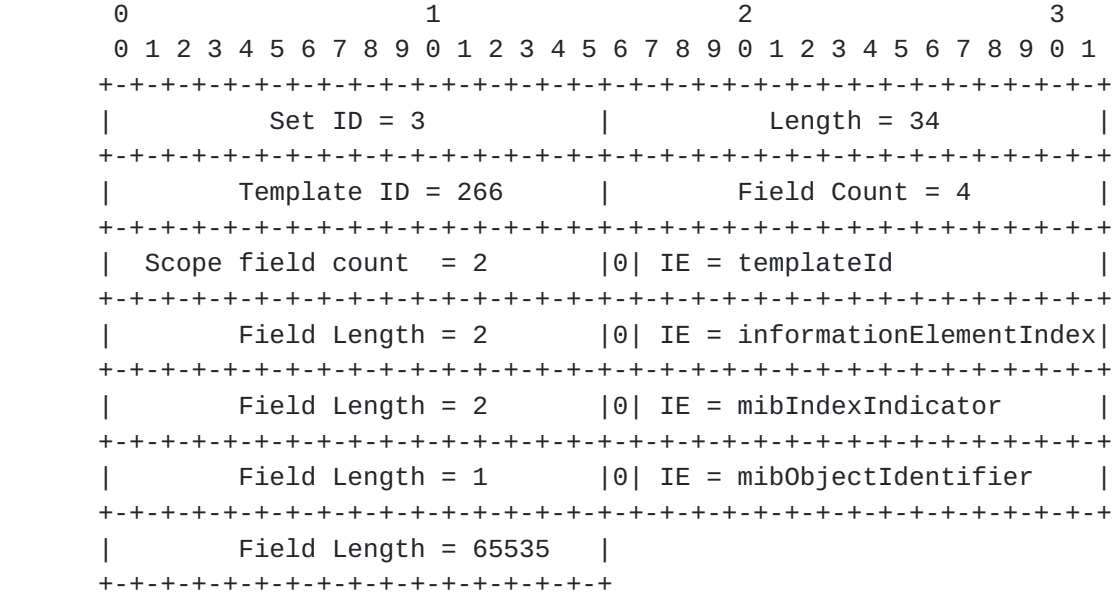

Figure 37: Example of mibFieldOption Template for a PSAMP Report with ifOutQLen indexed by a string index

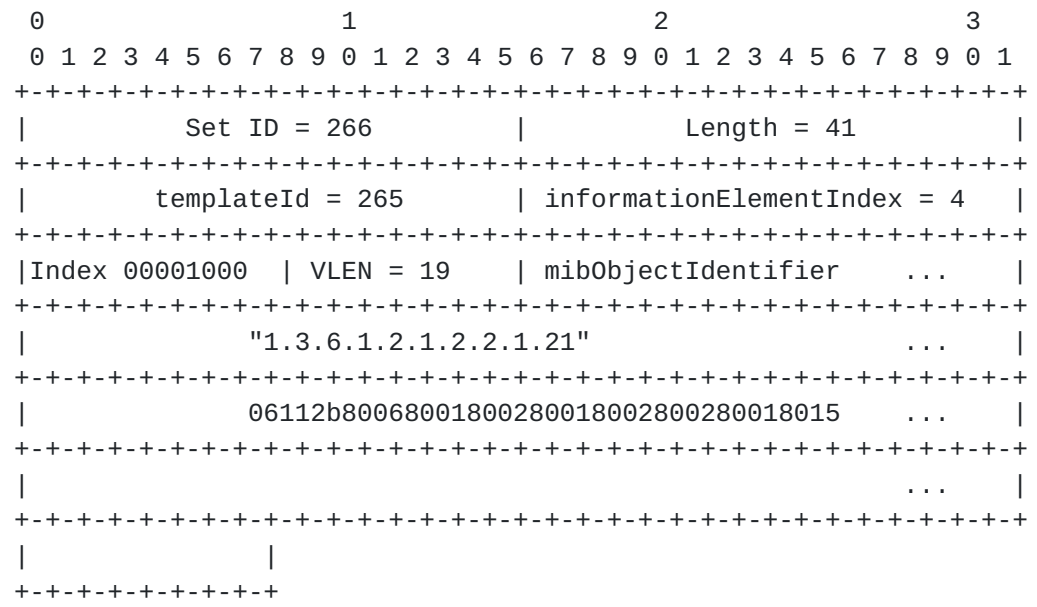

Figure 38: Example of mibFieldOption Data Record for a PSAMP Report with ifOutQLen indexed by a string index

Note that IPFIX reduced size encoding [RFC7011] 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 39. 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.

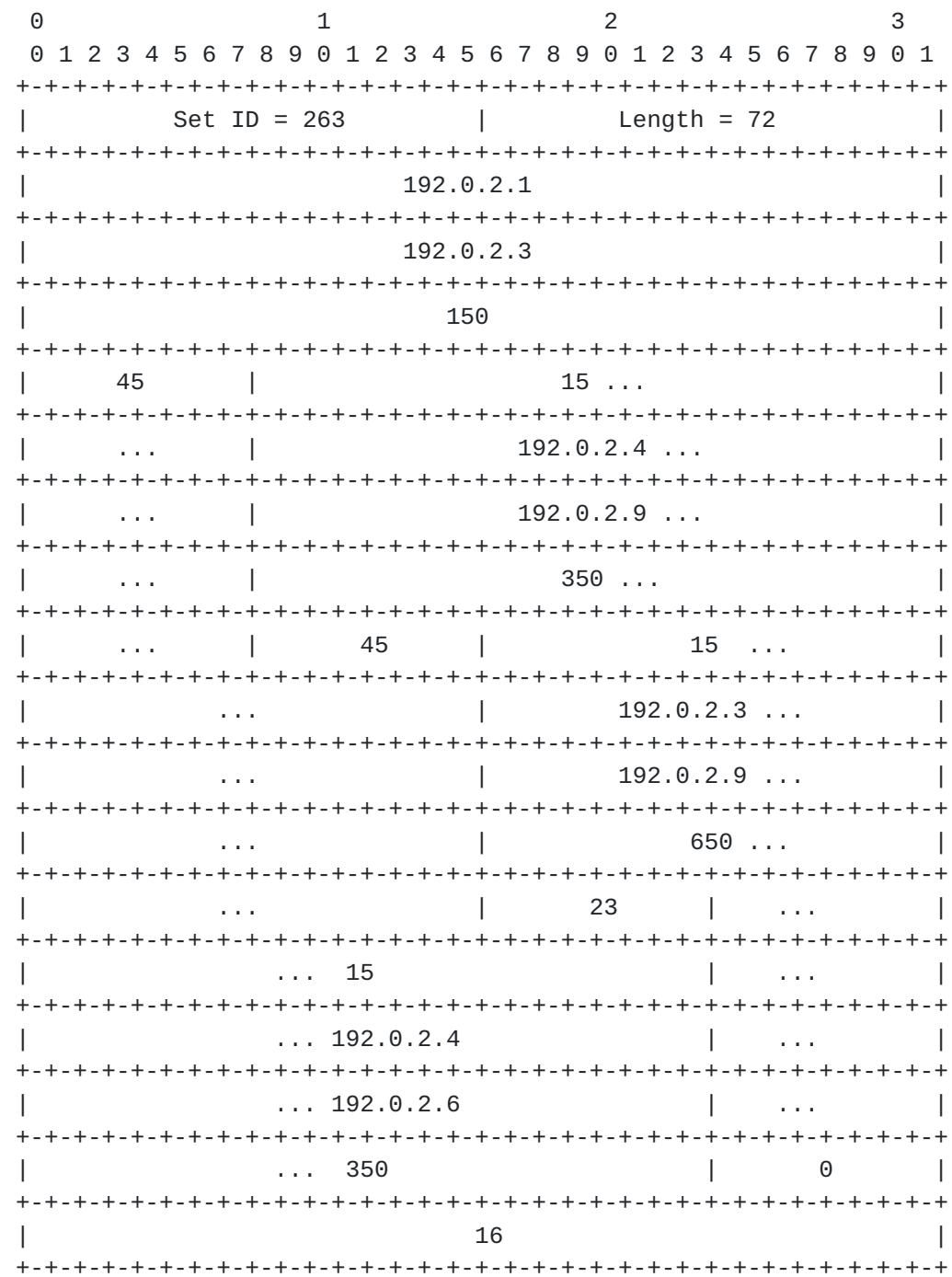

Figure 39: Example of PSAMP Packet Report with the ifOutQLen using ifIndex from IF-MIB [RFC2863] as an index

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 5.4.

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 [RFC7011].

The specifications in section 9 of [RFC7011] 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 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, 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 Identifier that it cannot decode, it SHOULD log an error.

If a Collecting Process receives a MIB Object Identifier 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 a 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 [RFC7011].

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 Data Records sent to certain collectors. Network operators should take care that the only MIB objects which are included in IPFIX Data Records are ones which the receiving flow collector is allowed to receive.

11. IANA Considerations

11.1. New Information Elements

The following new Information Elements must be allocated in IANA's IPFIX registry [IANA-IPFIX], as defined in the following sub-sections:

- mibObjectValueInteger
- mibObjectValueOctetString
- mibObjectValueOID
- mibObjectValueBITS
- mibObjectValueIPAddress
- mibObjectValueCounter
- mibObjectValueGauge
- mibObjectValueTime
- mibObjectValueUnsigned
- mibObjectValueTable
- mibObjectValueSequence
- mibSubIdentifier
- mibObjectIdentifier

- mibIndexIndicator
- mibCaptureTimeSemantics
- mibContextEngineID
- mibContextName
- mibObjectName
- mibObjectDescription
- mibObjectSyntax
- mibModuleName

11.1.1. New MIB Value Information Elements

11.1.1.1. mibObjectValueInteger

A new Information Element "mibObjectValueInteger" must be allocated in IANA's IPFIX registry [IANA-IPFIX], with the following definition:

Description: An IPFIX Information Element which denotes that an integer value from a MIB will be exported. The MIB Object Identifier ("mibObjectIdentifier") for this field MUST be exported in a MIB Field Option or via another means. This IE is used for MIB Objects with the Base Syntax of Integer32 and INTEGER with RLE used as required. The value is encoded as per the standard IPFIX Abstract Data Type of signed64.

Abstract Data Type: signed64

Data Type Semantics: default

ElementId: TBD1

Status: current

Reference: [this document].

11.1.1.2. mibObjectValueOctetString

A new Information Element "mibObjectValueOctetString" must be allocated in IANA's IPFIX registry [IANA-IPFIX], with the following definition:

Description: An IPFIX Information Element which denotes that an Octet String or Opaque value from a MIB will be exported. The MIB Object Identifier ("mibObjectIdentifier") for this field MUST be exported in a MIB Field Option or via another means. This IE is used for MIB Objects with the Base Syntax of OCTET STRING and Opaque. The value is encoded as per the standard IPFIX Abstract Data Type of octetArray.

Abstract Data Type: octetArray

ElementId: TBD2

Status: current

Reference: [this document].

11.1.1.3. mibObjectValueOID

A new Information Element "mibObjectValueOID" must be allocated in IANA's IPFIX registry [IANA-IPFIX], with the following definition:

Description: An IPFIX Information Element which denotes that an Object Identifier or OID value from a MIB will be exported. The MIB Object Identifier ("mibObjectIdentifier") for this field MUST be exported in a MIB Field Option or via another means. This IE is used for MIB Objects with the Base Syntax of OBJECT IDENTIFIER. Note - In this case the "mibObjectIdentifier" will define which MIB Object is being exported while the value contained in this IE will be an OID as a value. The mibObjectValueOID Information Element is encoded as ASN.1/BER [BER] in an octetArray.

Abstract Data Type: octetArray

ElementId: TBD3

Status: current

Reference: [this document].

11.1.1.4. mibObjectValueBITS

A new Information Element "mibObjectValueBITS" must be allocated in IANA's IPFIX registry [IANA-IPFIX], with the following definition:

Description: An IPFIX Information Element which denotes that a set of Enumerated flags or bits from a MIB will be exported. The MIB Object Identifier ("mibObjectIdentifier") for this field MUST be exported in a MIB Field Option or via another means. This IE is

used for MIB Objects with the Base Syntax of BITS. The value is encoded as per the standard IPFIX Abstract Data Type of unsigned64.

Abstract Data Type: unsigned64

Data Type Semantics: flags

ElementId: TBD4

Status: current

Reference: [this document].

11.1.1.5. mibObjectValueIPAddress

A new Information Element "mibObjectValueIPAddress" must be allocated in IANA's IPFIX registry [IANA-IPFIX], with the following definition:

Description: An IPFIX Information Element which denotes that an IPv4 Address Object from a MIB will be exported. The MIB Object Identifier ("mibObjectIdentifier") for this field MUST be exported in a MIB Field Option or via another means. This IE is used for MIB Objects with the Base Syntax of IPAddress. The value is encoded as per the standard IPFIX Abstract Data Type of ipv4Address.

Abstract Data Type: ipv4Address

ElementId: TBD5

Status: current

Reference: [this document].

11.1.1.6. mibObjectValueCounter

A new Information Element "mibObjectValueCounter" must be allocated in IANA's IPFIX registry [IANA-IPFIX], with the following definition:

Description: An IPFIX Information Element which denotes that a counter value from a MIB will be exported. The MIB Object Identifier ("mibObjectIdentifier") for this field MUST be exported in a MIB Field Option or via another means. This IE is used for MIB Objects with the Base Syntax of Counter32 or Counter64 with RLE used as required. The value is encoded as per the standard IPFIX Abstract Data Type of unsigned64.

Abstract Data Type: unsigned64

Data Type Semantics: totalCounter

ElementId: TBD6

Status: current

Reference: [this document].

11.1.1.7. mibObjectValueGauge

A new Information Element "mibObjectValueGauge" must be allocated in IANA's IPFIX registry [IANA-IPFIX], with the following definition:

Description: An IPFIX Information Element which denotes that a Gauge value from a MIB will be exported. The MIB Object Identifier ("mibObjectIdentifier") for this field MUST be exported in a MIB Field Option or via another means. This IE is used for MIB Objects with the Base Syntax of Gauge32. The value is encoded as per the standard IPFIX Abstract Data Type of unsigned64. This value will represent a non-negative integer, which may increase or decrease, but shall never exceed a maximum value, nor fall below a minum value.

Abstract Data Type: unsigned32

Data Type Semantics: totalCounter

ElementId: TBD7

Status: current

Reference: [this document].

11.1.1.8. mibObjectValueTime

A new Information Element "mibObjectValueTime" must be allocated in IANA's IPFIX registry [IANA-IPFIX], with the following definition:

Description: An IPFIX Information Element which denotes that a TimeTicks value from a MIB will be exported. The MIB Object Identifier ("mibObjectIdentifier") for this field MUST be exported in a MIB Field Option or via another means. This IE is used for MIB Objects with the Base Syntax of TimeTicks. The value is encoded as per the standard IPFIX Abstract Data Type of dateTimeMilliseconds.

Abstract Data Type: dateTimeMilliseconds

ElementId: TBD8

Status: current

Reference: [this document].

11.1.1.9. mibObjectValueUnsigned

A new Information Element "mibObjectValueUnsigned" must be allocated in IANA's IPFIX registry [IANA-IPFIX], with the following definition:

Description: An IPFIX Information Element which denotes that an unsigned integer value from a MIB will be exported. The MIB Object Identifier ("mibObjectIdentifier") for this field MUST be exported in a MIB Field Option or via another means. This IE is used for MIB Objects with the Base Syntax of unsigned64 with RLE used as required. The value is encoded as per the standard IPFIX Abstract Data Type of unsigned64.

Abstract Data Type: unsigned64

ElementId: TBD9

Status: current

Reference: [this document].

11.1.1.10. mibObjectValueTable

A new Information Element "mibObjectValueTable" must be allocated in IANA's IPFIX registry [IANA-IPFIX], with the following definition:

Description: An IPFIX Information Element which denotes that a complete MIB 'SEQUENCE OF X' or conceptual table value from a MIB will be exported. The MIB Object Identifier ("mibObjectIdentifier") for this field MUST be exported in a MIB Field Option or via another means. This IE is used for MIB Objects with a SYNTAX of SEQUENCE OF. This is encoded as a basicList of "mibObjectValueSequence"s. The mibObjectValueSequence MUST be of the same type/OID as specified in this MIB Object's syntax.

Abstract Data Type: basicList

ElementId: TBD10

Status: current

Reference: [this document].

11.1.1.11. mibObjectValueSequence

A new Information Element "mibObjectValueSequence" must be allocated in IANA's IPFIX registry [IANA-IPFIX], with the following definition:

Description: An IPFIX Information Element which denotes that a MIB SEQUENCE or a row from a conceptual table value as defined in a MIB will be exported. The MIB Object Identifier ("mibObjectIdentifier") for this field MUST be exported in a MIB Field Option or via another means. This IE is used for MIB Objects with a SYNTAX of SEQUENCE. This is encoded as a templateList of mibObjectValue<> IEs. The template specified in MUST be an Option Template and MUST include all the Objects listed in the INDEX clause as Scope fields.

Abstract Data Type: templateList

ElementId: TBD11

Status: current

Reference: [this document].

11.1.1.12. mibSubIdentifier

A new Information Element "mibSubIdentifier" must be allocated in IANA's IPFIX registry [IANA-IPFIX], with the following definition:

Description: A non-negative sub-identifier. One Sub number from an Object Identifier (OID). Can be used to provide an index to a columnar object without fully exporting the details of the Index.

Abstract Data Type: unsigned32

Data Type Semantics: identifier

ElementId: TBD12

Status: current

Reference: [this document].

11.1.2. New mibFieldOption Information Elements

11.1.2.1. mibObjectIdentifier

A new Information Element "mibObjectIdentifier" must be allocated in IANA's IPFIX registry [IANA-IPFIX], with the following definition:

Description: An IPFIX Information Element which denotes that a MIB Object Identifier (MIB OID) is exported in the (Options) Template Record. The mibObjectIdentifier Information Element contains the MIB OID encoded as ASN.1/BER [BER]

Abstract Data Type: octetArray

ElementId: TBD13

Status: current

Reference: [this document].

11.1.2.2. mibIndexIndicator

A new Information Element "mibIndexIndicator" must be allocated in IANA's IPFIX registry [IANA-IPFIX], with the following definition:

Description: This set of bit fields is used for marking the Information Elements of a Data Record that serve as Index Objects or IE for a mibField. Each bit represents an Information Element in the Data Record with the n-th bit representing the n-th Information Element. A bit set to value 1 indicates that the corresponding Information Element is an Index of the Columner Object represented by the mibFieldValue. A bit set to value 0 indicates that this is not the case. If the Data Record contains more than 64 Information Elements, the corresponding Template SHOULD be designed such that all Index Fields are among the first 64 Information Elements, because the mibIndexIndicator only contains 64 bits. If the Data Record contains less than 64 Information Elements, then the bits in the flowKeyIndicator for which no corresponding Information Element exists MUST have the value 0.

Abstract Data Type: unsigned64

Data Type Semantics: flags

ElementId: TBD14

Status: current

Reference: [this document].

11.1.2.3. mibCaptureTimeSemantics

A new Information Element "mibCaptureTimeSemantics" must be allocated in IANA's IPFIX registry [IANA-IPFIX], with the following definition:

Description: Indicates when in the lifetime of the flow the MIB value was extracted for a mibObjectIdentifier. This is used to indicate if the value exported was collected from the MIB closer to flow creation or flow export time and will refer to the Timestamp fields included in the same record. This field SHOULD be used when exporting mibObjectValues that specify counters or statistics.

Values:

1. undefined
2. flow creation - The value for the MIB Object is captured from the MIB when the Flow is created
3. flow end - The value for the MIB Object is captured from the MIB when the Flow ends
4. flow export - The value for the MIB Object is captured from the MIB the Flow is being Exported
5. average - The value for the MIB Object is an average of multiple captures from the MIB over the life of the Flow

Abstract Data Type: unsigned8

Data Type Semantics: identifier

ElementId: TBD15

Status: current

Reference: [this document].

11.1.2.4. mibContextEngineID

A new Information Element "mibContextEngineID" must be allocated in IANA's IPFIX registry [IANA-IPFIX], with the following definition:

Description: A mibContextEngineID (also known as snmpEngineID) that sepcifies the SNMP engine ID for a mib field being exported over IPFIX. Definition as per [RFC3411] section 3.3

Abstract Data Type: octetArray

ElementId: TBD16

Status: current

Reference: [this document].

11.1.2.5. mibContextName

A new Information Element "mibContextName" must be allocated in IANA's IPFIX registry [IANA-IPFIX], with the following definition:

Description: This Information Element denotes that a MIB Context Name is specified for a mib field being exported over IPFIX. Reference [RFC3411] section 3.3

Abstract Data Type: string

ElementId: TBD17

Status: current

Reference: [this document].

11.1.3. New mibType Information Elements

11.1.3.1. mibObjectName

A new Information Element "mibObjectName" must be allocated in IANA's IPFIX registry [IANA-IPFIX], with the following definition:

Description: The name (called a descriptor in RFC 2578) of an object type definition.

Abstract Data Type: string

ElementId: TBD18

Status: current

Reference: [this document].

11.1.3.2. mibObjectDescription

A new Information Element "mibObjectDescription" must be allocated in IANA's IPFIX registry [IANA-IPFIX], with the following definition:

Description: The value of the DESCRIPTION clause of an MIB object type definition.

Abstract Data Type: string

ElementId: TBD19

Status: current

Reference: [this document].

11.1.3.3. mibObjectSyntax

A new Information Element "mibObjectSyntax" must be allocated in IANA's IPFIX registry [IANA-IPFIX], with the following definition:

Description: The value of the SYNTAX clause of an MIB object type definition. This will start with abstract data structure of the MIB object which may be a base type, the BITS construct, a textual convention, SEQUENCE OF or SEQUENCE. This may be followed by subtyping or enumeration for the object. [RFC2578]

Abstract Data Type: string

ElementId: TBD20

Status: current

Reference: [this document].

11.1.3.4. mibModuleName

A new Information Element "mibModuleName" must be allocated in IANA's IPFIX registry [IANA-IPFIX], with the following definition:

Description: The textual name of the MIB that defines a MIB OID Object.

Abstract Data Type: string

ElementId: TBD21

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

[BER] International Organization for Standardization, "Information processing systems - Open Systems Interconnection - Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1),International Organization for Standardization. International Standard 8825,", <http://www.iso.org>.

[IANA-DATATYPES] IANA, "IPFIX Information Element Data Types registry", , <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>.

[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.

[RFC3411] Harrington, D., Presuhn, R., and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", STD 62, RFC 3411, December 2002.

[RFC4293] Routhier, S., "Management Information Base for the Internet Protocol (IP)", RFC 4293, April 2006.

[RFC7011] Claise, B., Trammell, B., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, September 2013.

[RFC7012] Claise, B. and B. Trammell, "Information Model for IP Flow Information Export (IPFIX)", RFC 7012, September 2013.

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

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

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

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

Phone: +32 2 704 5622
Email: bclaise@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

Colin McDowall
Cisco Systems, Inc.
96 Commercial Quay
Commercial Street
Edinburgh EH6 6LX
UK

Phone: +44 131 561 3634
Email: cmcdowal@cisco.com

Juergen Schoenwaelder
Jacobs University Bremen
Campus Ring 1
Bremen 28725
Germany

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