Internet Engineering Task Force (IETF)
Request for Comments: 8038
Category: Standards Track
ISSN: 2070-1721

P. Aitken, Ed.
Brocade
B. Claise
Cisco Systems, Inc.
S. B S
Mojo Networks, Inc.
C. McDowall
Brocade
J. Schoenwaelder
Jacobs University Bremen

May 2017

Exporting MIB Variables Using the IP Flow Information Export (IPFIX) Protocol

Abstract

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

Two IPFIX Options Templates, as well as a method for creating IPFIX Options Templates that are used to export the extra data required to fully describe Simple Network Management Protocol (SNMP) MIB objects in IPFIX, are specified herein.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc8038.

Table of Contents

1. Introduction 4
2. Motivation 5
3. Terminology 7
4. High-Level Solution Overview 8
5. MIB Object Value Information Elements and the MIB Field Options Template 10
5.1. MIB Field Options Architecture 11
5.2. IPFIX and MIB Data Model 13
5.3. MIB Field Options - Specifications and Required Fields 15
5.3.1. MIB Field Options Template 16
5.3.2. MIB Type Options Template 16
5.4. MIB Field Options Template Formats 17
5.4.1. Data Template Containing a mibObjectValue Field 17
5.4.2. MIB Field Options Template 19
5.4.3. MIB Field Options Data Records 20
5.4.4. Options Template Containing a mibObjectValue Field 21
5.4.5. MIB Field Options Template with Semantics Fields 23
5.4.6. MIB Field Options Template with Extra MIB Object Details 24
5.5. Use of Field Order in the MIB Field Options Template 27
5.6. Identifying the SNMP Context 27
5.7. Template Management 28
5.7.1. Large Messages 28
5.7.2. Template Withdrawal and Reuse 29

5.8. Exporting Conceptual Rows and Tables 29
5.8.1. Exporting Conceptual Rows - Indexing 30
5.8.2. Exporting Conceptual Rows - mibObjectValueRow 30
5.8.3. Exporting Conceptual Rows - AUGMENTS 36
5.8.4. Exporting Conceptual Tables - mibObjectValueTable ..37
5.8.5. Exporting Columnar Objects: Using mibIndexIndicator 38
6. Example Use Cases 39
6.1. Non-columnar MIB Object: Established TCP Connections 39
6.2. Enterprise-Specific MIB Object: Detailing CPU Load History 42
6.3. Exporting a Conceptual Row: The OSPF Neighbor Row 45
6.4. Exporting Augmented Conceptual Row: Mapping IF-MIB ID to Name 49
6.5. Exporting a Columnar Object: ipIfStatsInForwDatagrams 55
6.6. Exporting a Columnar Object Indexed by Information Elements: ifOutQLen 58
6.7. Exporting with Multiple Contexts: The OSPF Neighbor Row Revisited 62
7. Configuration Considerations 65
8. The Collecting Process's Side 66
9. Applicability 66
10. Security Considerations 67
11. IANA Considerations 68
11.1. New IPFIX Semantics 68
11.1.1. snmpCounter 68
11.1.2. snmpGauge 68
11.2. New IPFIX Information Elements 69
11.2.1. New MIB Object Value Information Elements 69
11.2.2. New MIB Field Options Information Elements 75
11.2.3. New MIB Type Information Elements 79
12. References 81
12.1. Normative References 81
12.2. Informative References 82
Acknowledgments 84
Authors' Addresses 84

1. Introduction

There is growing interest in using IP Flow Information Export (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 [IANA-IPFIX], Information Elements coming from the SNMP world have already been specified, e.g., ingressInterface and egressInterface both refer to the ifIndex object as defined in [RFC2863].

In particular, the Management Information Base was designed as a separate system of definitions; this opens up the possibility of 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.

Therefore, the primary goals of this document are:

- to specify a way to complement IPFIX Data Records with MIB objects;
- to avoid the need to define new IPFIX Information Elements for existing MIB objects that are already fully specified;
- to allow the correlation of SNMP and IPFIX sourced data by exporting them together; and
- to allow SNMP push data from SNMP-only devices to be more easily integrated into IPFIX-based collection infrastructures.

2. Motivation

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 Data Records with useful and already-standardized information. Special consideration is given to the case of an existing Template Record that needs to be augmented with some MIB variables whose index is already present in the Template Record as an IPFIX Information Element -- for example, a 7-tuple Data Record containing the ingressInterface Information Element, which needs to be augmented by interface counters [RFC2863] that are indexed by the respective ingressInterface values already contained in the Data Records. See Section 3 for terminology definitions.

Many Data Records contain the ingressInterface and/or the egressInterface Information Elements. 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 because counters carried in Data 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 an SNMP polling engine, SNMP data is retrieved shortly after Data Records have been received, which implies a delay of the sum of the active or idle timeouts (if not null) plus the time to export the Data 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 significantly higher. See [RFC5102] for details regarding active and idle timeouts.

This document does not specify how to carry SNMP notifications in IPFIX, 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 new textual conventions, can be exported within IPFIX and then decoded in the Collector. However, since a textual convention can contain almost any name, this document does not extend the existing "IPFIX Information Elements" subregistry [IANA-IPFIX] that contains informationElementDataType.

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

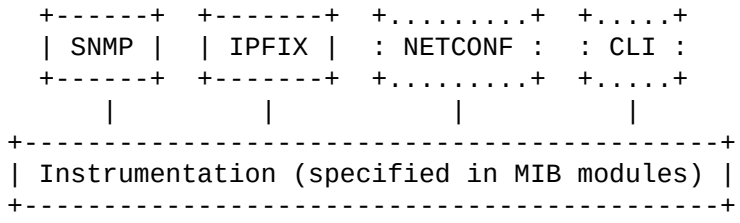


Figure 1: Architectural Model

3. Terminology

IPFIX-specific terminology (Information Element, Template, Template Record, Options Template Record, Template Set, Collector, Exporter, Data Record, Transport Session, Exporting Process, Collecting Process, 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") in relation to the export of MIB objects.

Object Identifier (MIB OID)

An Object Identifier value is an ordered list of non-negative numbers. For SMIv2, each number in the list is referred to as a sub-identifier. There are at most 128 sub-identifiers in a value, and each sub-identifier has a maximum value of 2^32 - 1 (4294967295 decimal). See [RFC2578], Section 3.5.

MIB Object Identifier Information Element

An IPFIX Information Element ("mibObjectIdentifier") that denotes that a MIB Object Identifier (MIB OID) is exported in the (Options) Data Record. See Section 11.2.2.1.

SMIv2 Terminology

The key words "MIB module", "MIB object", "INDEX", "AUGMENTS", "textual convention", "columnar object", "conceptual row", and "conceptual table" in this document are to be interpreted as described in SMIv2 [RFC2578].

SMIv2 SYNTAX

The SYNTAX key words "INTEGER", "Integer32", "OCTET STRING", "OBJECT IDENTIFIER", "BITS", "IpAddress", "Counter32", "Gauge32", "TimeTicks", "Opaque", "Counter64", "Unsigned32", "SEQUENCE", and "SEQUENCE OF" in this document are to be interpreted as described in SMIv2 [RFC2578].

SNMP Context Terminology

The key words "snmpEngineID", "contextEngineID", and "contextName" in this document are to be interpreted as described in [RFC3411].

mibObjectValue Information Elements

"mibObjectValue Information Elements" refers to any and all of the mibObjectValue Information Elements generically. Any restriction or requirement in this document that refers to "mibObjectValue" applies to the following Information Elements as defined in Section 11.2.1: mibObjectValueInteger, mibObjectValueOctetString, mibObjectValueOID, mibObjectValueBits, mibObjectValueIPAddress, mibObjectValueCounter, mibObjectValueGauge, mibObjectValueTimeTicks, mibObjectValueUnsigned, mibObjectValueTable, and mibObjectValueRow.

Abstract Data Type

Abstract Data Types for IPFIX are defined in Section 3.1 of [RFC7012]. This specification uses the Abstract Data Types "unsigned8", "unsigned32", "unsigned64", "signed32", "octetArray", "string", "ipv4Address", and "subTemplateList".

IE

Used as shorthand for "Information Element" [RFC7011] in the figures.

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].

4. High-Level Solution Overview

This document specifies a method for creating IPFIX Options Templates that are used to export the extra data required to describe MIB variables (see Section 5.1).

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-columnar or columnar MIB object(s). Enterprise-specific MIB Object Identifiers are also supported.

This document also defines two standard IPFIX Options Templates (see Section 5.3) that are used as part of the mechanism to export MIB object metadata:

- MIB Field Options Template (Section 5.3.1)
- MIB Type Options Template (Section 5.3.2)

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:

- MIB Object Value Information Elements (Section 11.2.1)
- MIB Field Options Information Elements (Section 11.2.2)
- MIB Type Information Elements (Section 11.2.3)

Additionally, this document defines two new IPFIX semantics that are required for the new Information Elements:

- snmpCounter (Section 11.1.1)
- snmpGauge (Section 11.1.2)

Two common types defined in SMIv2 are conceptual rows and conceptual tables. It is desirable that exporting a complete or partial conceptual row be simple and efficient. This is accomplished by using IPFIX Structured Data [RFC6313] to reduce repetition of Object Identifier and indexing data.

To allow the use of individual columnar objects that make up a conceptual row, a method is also specified to explain that a MIB object is indexed by other fields in the same Data Flow. For an individually indexed mibObjectValue, the index fields are sent in the same way as any of the other fields in the same Data Record and may be mibObjectValue Information Element(s) or other existing Information Element(s).

Also, in some cases Exporters may not want (or be able) to export the full information on how the MIB objects being exported are indexed. This may be because the MIB object is being used purely as type information or the Exporting Process may not have knowledge of the indexing required. Therefore, providing index information for columnar objects is optional.

5. MIB Object Value Information Elements and the MIB Field Options Template

This document defines new mibObjectValue Information Elements (in Section 11.2.1). These are used to export MIB objects as part of standard IPFIX Templates. The mibObjectValue Information Elements contain the actual data values.

The Metering Process or Exporting Process may extract the data values for mibObjectValue Information Elements from a Process that resides on the same device or may capture or create the data required to match the definition of the MIB object. In particular, exporting a value of a MIB object defined in a certain MIB module does not imply that the SNMP process on the device supports that MIB module.

The main issue that arises from exporting values of MIB objects in IPFIX is that MIB Object Identifiers do not fit into the standard IPFIX Template format [RFC7011], as this only provides a 16-bit Information Element identifier.

The values of a MIB object could be exported using a MIB-specific Information Element, without providing any Object Identifiers. However, without exporting the actual MIB OID, the full type of the data would be unknown, and every field containing MIB object data would appear identical. Without knowing which OID the contents of a field map to, the data would be incomprehensible to a Collector.

For the values in the mibObjectValue Information Elements to be understandable, more meta-information about the mibObjectValue Information Elements must be sent as part of the IPFIX export. The required minimum information to understand each field that is being exported is provided in Section 5.3.1.

One approach to this problem would be to extend the IPFIX standard to allow extended Field Specifiers so that metadata about fields can be included in Data Templates. This would, however, require a new version of the IPFIX standard that may not be backward compatible. However, future versions of IPFIX may export the required MIB metadata as part of newly defined IPFIX Set versions.

This document defines a MIB Field Options Template to export the extra meta-information required for mibObjectValue Information Elements. This is a standard IPFIX Options Template Set that includes a minimum set of required fields (see Section 5.3.1) and may include extra fields to provide more meta-information about one of the mibObjectValue Information Elements.

The MIB Field Options export tells the Collecting Process the OID for the MIB object type definition for the following (Template, field).

5.1. MIB Field Options Architecture

Four IPFIX Sets are used together to export a Flow that contains mibObjectValue Information Elements. These are:

1. A Template Set that includes the mibObjectValue Information Element.

   The Template Set informs the Collector that a MIB object value of length N will be exported. This Set may also be an Options Template Set.

2. A MIB Field Options Template Set.

   The MIB Field Options Template describes which metadata will be sent for each mibObjectValue Information Element being exported.

3. A MIB Field Options Data Set.

   The MIB Field Options Data Set includes the metadata for each MIB object (i.e., the mibObjectIdentifier or mibSubIdentifier). The metadata about the mibObjectValue Information Elements only needs to be resent as per normal Template refreshes or resends.

4. A Data Set.

   The Data Set contains only the actual data extracted from the MIB or described by the MIB module.

Figure 2 shows the IPFIX Message structure for a MIB field in a Template Set.

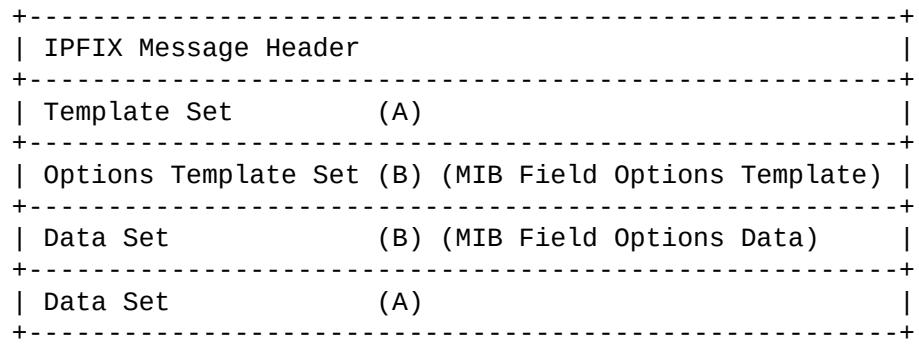


Figure 2: IPFIX Message Structure for a MIB Field in a Template Set

The MIB Field Options Template defines MIB Field Options Data Records. The MIB Field Options Data Records annotate the Data Template with mibObjectValue metadata. Together, the Data Template and MIB Field Options Data Records define the Data Records that will be exported.

The Data Records (A) have a dependency on the two Templates and the MIB Field Options Data Records.

More Data Sets that use the same mibObjectValue Information Element can then be sent in subsequent packets.

Figure 3 shows the relationships between the Sets discussed above.

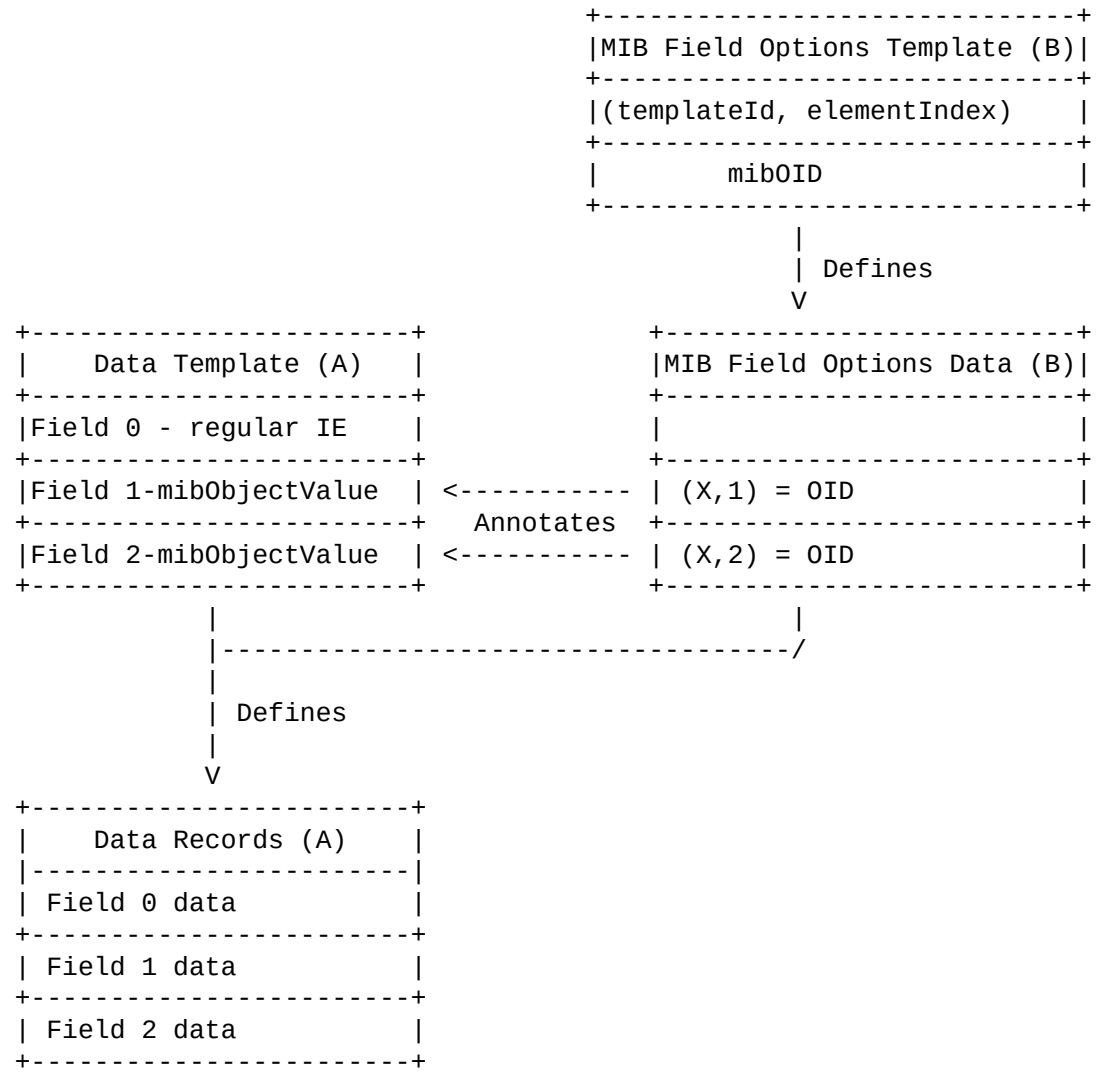


Figure 3: Relationships between Sets

5.2. IPFIX and MIB Data Model

[RFC2578], Section 7.1 specifies that the SYNTAX clause for a MIB object defines the abstract data structure of an object and what it must contain:

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

For each of the SYNTAX clause options, this document specifies exactly which mibObjectValue Information Element 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 sub-typing (e.g., integerSubType, octetStringSubType) [RFC2578], the mibObjectSyntax Information Element SHOULD be used to export this detail to the Collecting Process.

The options for the SYNTAX clause are then mapped as follows:

Section in RFC 2578	SYNTAX	mibObjectValue Information Element
7.1.1	INTEGER/Integer32	mibObjectValueInteger
7.1.2	OCTET STRING	mibObjectValueOctetString
7.1.3	OBJECT IDENTIFIER	mibObjectValueOID
7.1.4	BITS	mibObjectValueBits
7.1.5	IpAddress	mibObjectValueIPAddress
7.1.6	Counter32	mibObjectValueCounter
7.1.7	Gauge32	mibObjectValueGauge
7.1.8	TimeTicks	mibObjectValueTimeTicks
7.1.9	Opaque	mibObjectValueOctetString
7.1.10	Counter64	mibObjectValueCounter
7.1.11	Unsigned32	mibObjectValueUnsigned
7.1.12	SEQUENCE	mibObjectValueRow
7.1.12	SEQUENCE OF	mibObjectValueTable

Table 1: SMIv2 SYNTAX to mibObjectValue Types

Values are encoded as per the standard IPFIX encoding of Abstract Data Types. The only new encoding reference in this document is that Object Identifiers (OIDs) will be encoded as per ASN.1/BER [X.690] in an octetArray.

The mibObjectValue and mibObjectIdentifier Information Elements are standard IPFIX fields. Therefore, the E bit of the mibObjectValue or mibObjectIdentifier Information Elements is set to 0.

The MIB object being exported may be defined in an enterprise-specific MIB module, but the Information Elements defined in this standard are still exported with the E bit set to 0. The OID being exported indicates that the MIB object was defined in an enterprise-specific MIB module.

5.3. MIB Field Options - Specifications and Required Fields

For each mibObjectValue Information Element that is defined in an IPFIX Template, a MIB Field Options Data Record will be exported that provides the required minimum information to define the MIB object that is being exported (see Section 5.3.1).

The MIB Field Options Data Records are defined in a Template referred to in this document as a MIB Field Options Template with the format specified in Section 5.4.

The MIB Field Options Template and MIB Field Options Data Records MUST be exported in the same IPFIX Message as any Template that is using a mibObjectValue Information Element. Note that this places an implicit size constraint on the export.

This whole set of Templates and MIB Field Options Data Records MUST all be exported prior to the corresponding Data Records that depend upon them. That is, the export order MUST be:

1. Data Template for mibObjectValue Information Elements (Set ID 2)

2. MIB Field Options Template (Set ID 3)

3. MIB Field Options Data Records (Set ID >= 256)

4. MIB Object Value Data Records (Set ID >= 256)

Note that the ID of an identical MIB Field Options Template that has already been exported MAY be reused without exporting the Template again.

IPFIX Set IDs are defined in Section 3.3.2 of [RFC7011]. A value of 2 indicates a Template Set, a value of 3 indicates an Options Template Set, and values 256 and above indicate Data Sets.

5.3.1. MIB Field Options Template

Three fields are REQUIRED to unambiguously export a standalone mibObjectValue Information Element with a MIB Field Options Template:

- (scope) templateId [IANA-IPFIX]
- (scope) informationElementIndex [IANA-IPFIX]
- mibObjectIdentifier (Section 11.2.2.1) or mibSubIdentifier (Section 11.2.2.2)

These are the minimum fields required in a MIB Field Options Template (see Section 5.4.2).

The mibObjectIdentifier is used to provide the OID for all mibObjectValue Information Elements exported, except when IPFIX Structured Data [RFC6313] is being used to export a conceptual row (see Section 5.8.2).

While the following are optional, they are nevertheless RECOMMENDED in certain circumstances, as described in the referenced sections:

- mibCaptureTimeSemantics
  (discussed in Section 5.4.5; Information Element defined in Section 11.2.2.4)
- mibIndexIndicator
  (discussed in Section 5.8.5; Information Element defined in Section 11.2.2.3)
- mibContextEngineID
  (discussed in Section 5.6; Information Element defined in Section 11.2.2.5)
- mibContextName
  (discussed in Section 5.6; Information Element defined in Section 11.2.2.6)

5.3.2. MIB Type Options Template

There are also fields that provide type information from a MIB object definition that MAY be exported to a Collecting Process.

Type information is statically defined in a MIB module; it is not expected to change. However, the additional information about the MIB object may help a Collecting Process that does not have access to the MIB module.

To export a MIB Type Options Template, the mibObjectIdentifier is RECOMMENDED as a Scope Field so that it matches the MIB Field Options Template. Any combination of the other MIB Type fields may be included.

- (scope) mibObjectIdentifier (see Section 11.2.2.1)
- mibObjectName (see Section 11.2.3.1)
- mibObjectDescription (see Section 11.2.3.2)
- mibObjectSyntax (see Section 11.2.3.3)
- mibModuleName (see Section 11.2.3.4)

5.4. MIB Field Options Template Formats

5.4.1. Data Template Containing a mibObjectValue Field

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

The only extra requirement on a Template Record using one or more mibObjectValue Information Elements is that it MUST export the required metadata specified in Section 5.3.1 for EACH mibObjectValue Information Element.

If multiple MIB Field Options Data Records that refer to a mibObjectValue are received, the latest MUST be used. This matches the expected behavior of IPFIX Templates.

There is a one-to-one mapping between each mibObjectValue Information Element and a MIB Field Options Data Record.

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

Figure 4 shows an IPFIX Template Set using a mibObjectValue Information Element.

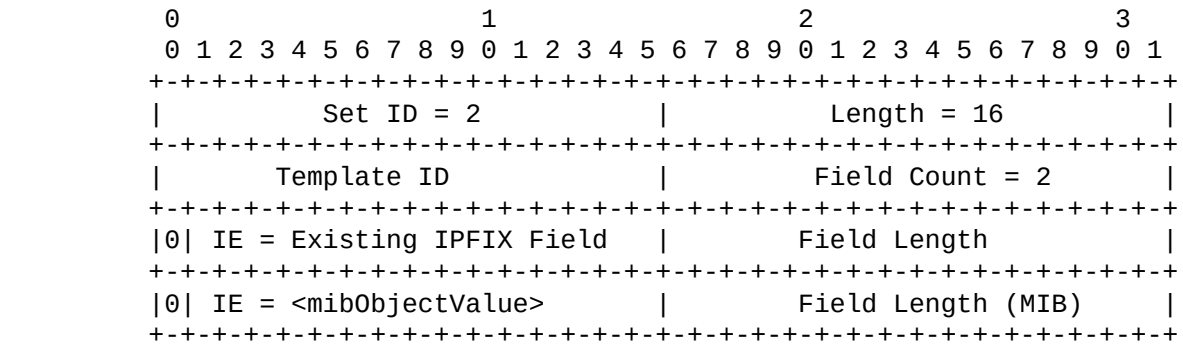


Figure 4: IPFIX Template Set Using mibObjectValue Information Element

Where:

<mibObjectValue>

One of the mibObjectValue IPFIX Information Elements that denotes that MIB object data (i.e., the value of a MIB object) will be exported in the (Options) Data Record.

This could be any one of the mibObjectValue Information Elements defined in Section 11.2.1: mibObjectValueInteger, mibObjectValueOctetString, mibObjectValueOID, mibObjectValueBits, mibObjectValueIPAddress, mibObjectValueCounter, mibObjectValueGauge, mibObjectValueTimeTicks, mibObjectValueUnsigned, mibObjectValueTable, and mibObjectValueRow.

When a mibObjectValue Information Element is used, the MIB Object Identifier ("mibObjectIdentifier") MUST be exported via a MIB Field Options Template and MIB Field Options Data Record. See Section 5.3.1.

Field Length (MIB)

The length of the encoded MIB object data in the corresponding Data Records, in octets. See [RFC7011] for a detailed definition. 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].

5.4.2. MIB Field Options Template

The MIB Field Options Template is a standard Options Template that defines the fields that will be exported to provide enough metadata about a mibObjectValue Information Element so that the Collector can tie the data values in the mibObjectValue Information Element back to the definition of the MIB object.

All MIB Field Options Templates contain the fields specified in Section 5.3.1.

Figure 5 shows the required fields to export a mibObjectIdentifier for the MIB Field Options Template format.

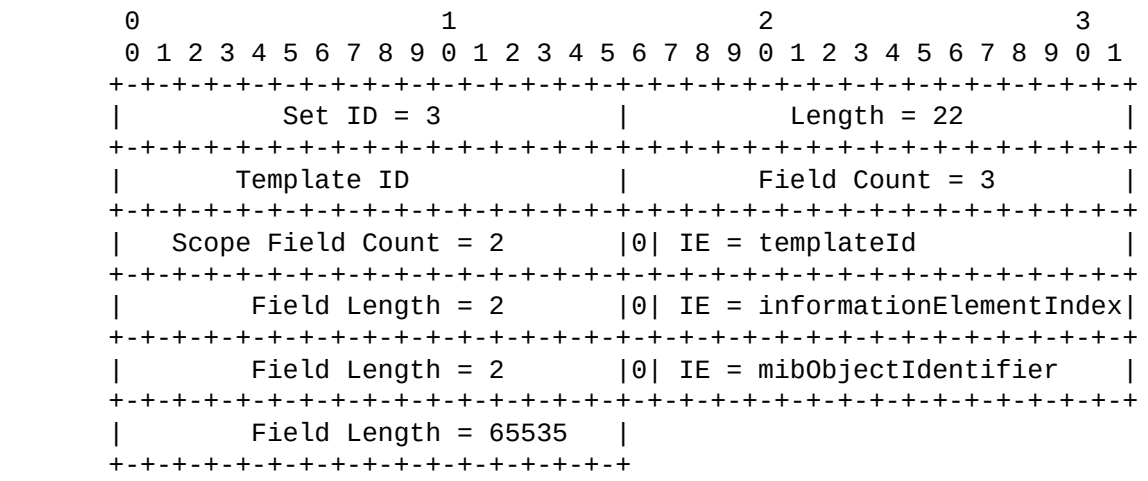


Figure 5: MIB Field Options Template Format - Required Fields

Where:

templateId

The first Scope Field is an IPFIX Information Element that denotes that a Template Identifier will be exported as part of the MIB Field Options Data Record. This Template Identifier, paired with an index into that Template (the "informationElementIndex" field), uniquely references one mibObjectValue Information Element being exported.

informationElementIndex

The second Scope Field is an IPFIX Information Element that denotes a zero-based index into the fields defined by a Template. When paired with a "templateId", this uniquely references one mibObjectValue Information Element being exported.

mibObjectIdentifier

An IPFIX Information Element that denotes the MIB Object Identifier for the mibObjectValue Information Element exported in the (Options) Template Record.

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

The Object Identifier is encoded in the IPFIX Data Record in ASN.1/BER [X.690] format.

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

Variable-length encoding is indicated by the Field Length value of 65535, per Sections 3.2 and 7 of [RFC7011]. The RECOMMENDED use of variable-length encoding for mibObjectIdentifier fields is indicated in subsequent figures by placing 65535 in the relevant length fields.

5.4.3. MIB Field Options Data Records

The MIB Field Options Data Records conform to the Template Specification in the MIB Field Options Template. There may be multiple MIB Field Options Data Records exported.

The Collecting Process MUST store all received MIB Field Options Data information for the duration of each Transport Session, because the Collecting Process will need to refer to the extra meta-information to fully decode each mibObjectValue Information Element.

Figure 6 shows the format of the exported MIB Field Options Data Record, detailing the metadata that will be exported to match the Template in Figure 5.

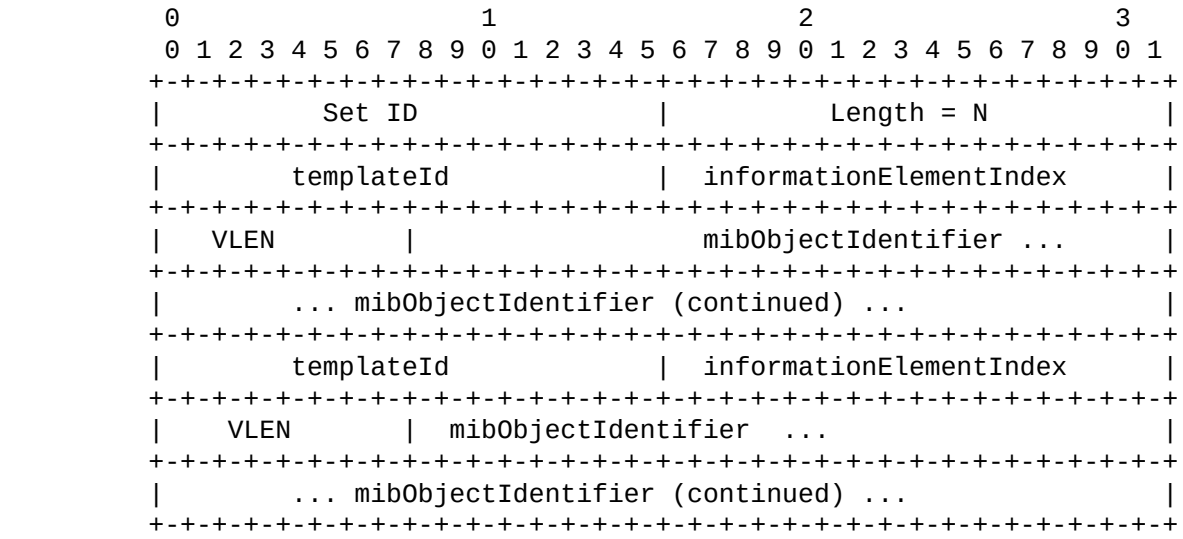


Figure 6: Format of MIB Field Options Data Record

VLEN contains the variable length of the mibObjectIdentifier per Section 7 of [RFC7011].

5.4.4. Options Template Containing a mibObjectValue Field

The Options Template Record format of a Template that uses a mibObjectValue Information Element is identical to the standard format as defined in [RFC7011]. The mibObjectValue Information Element is specified using standard Field Specifiers per [RFC7011].

A mibObjectValue Information Element can be either a Scope Field or a non-Scope Field in an Options Template Record.

The only extra requirement on an Options Template Record using one or more mibObjectValue Information Elements is that it MUST export the required metadata specified in Section 5.3.1 for EACH mibObjectValue Information Element.

An IPFIX Options Template Record MUST export a MIB Field Options Template and Data Record to provide the minimum required metadata for each mibObjectValue Information Element.

Figure 7 shows an IPFIX Options Template Set using an existing IPFIX field as a Scope Field and with a mibObjectValueInteger Information Element as a non-Scope Field, while Figure 8 shows an IPFIX Options Template Set using a mibObjectValueInteger Information Element as a Scope Field with an existing IPFIX field as a non-Scope Field.

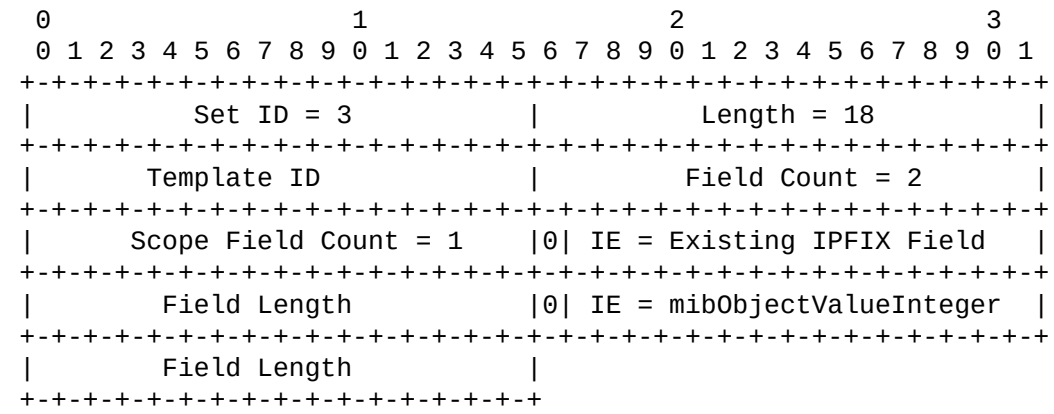


Figure 7: IPFIX Options Template Set Using a Non-Scope mibObjectValueInteger Field

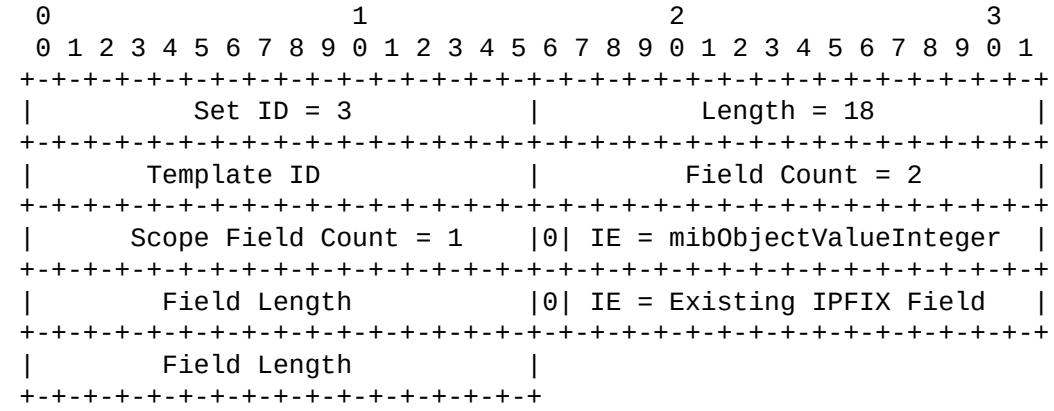


Figure 8: IPFIX Options Template Set Using a Scope mibObjectValueInteger Field

5.4.5. MIB Field Options Template with Semantics Fields

A MIB Field Options Template MAY specify that extra Information Elements will be exported to record how the mibObjectValue was collected.

Alternatively, one of the existing IPFIX observationTime* elements [IANA-IPFIX] may be exported to specify exactly when the value was collected.

Figure 9 shows the MIB Field Options Template for a non-columnar field with Semantic Data.

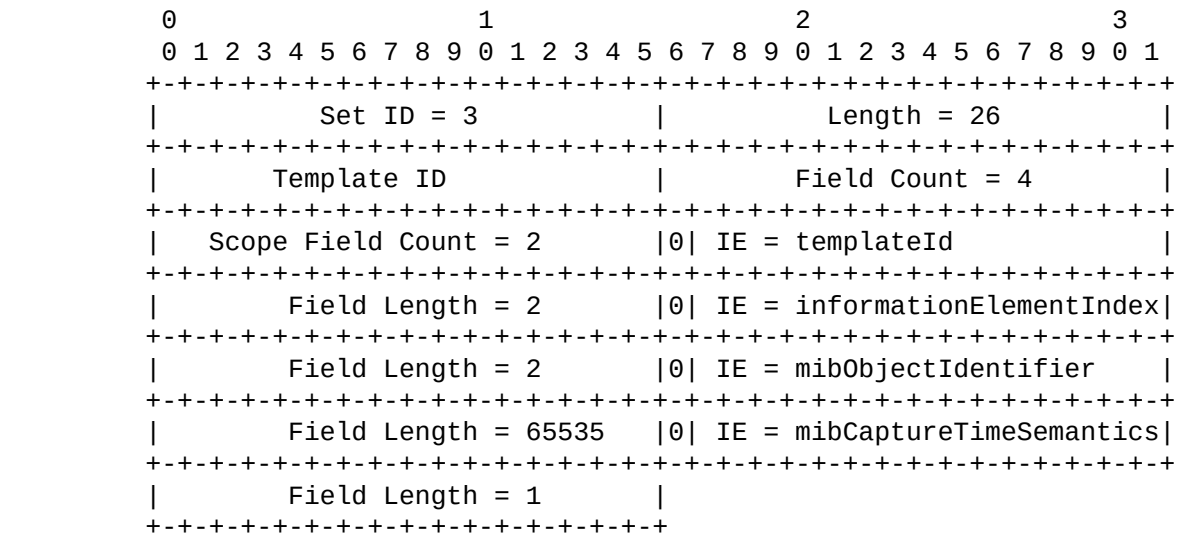


Figure 9: MIB Field Options Template for a Non-columnar Field with Semantic Data

Where:

mibObjectIdentifier

Note the use of variable-length encoding for this field.

mibCaptureTimeSemantics

The MIB Capture Time Semantics IPFIX Information Element, as defined in Section 11.2.2.4.

It is RECOMMENDED to include this field when exporting a mibObjectValue Information Element that specifies counters or statistics, particularly for situations with long-lived Flows.

5.4.6. MIB Field Options Template with Extra MIB Object Details

The OID exported within the mibObjectIdentifier IPFIX Information Element provides an OID reference to a MIB object type definition that will fully describe the MIB object data being exported.

However, an Exporting Process MAY decide to include some extra fields to more fully describe the MIB object that is being exported with a mibObjectValue Information Element.

This can be helpful if the Collecting Process may not have access to the MIB module.

The Exporting Process can either include the fields with extra object details as part of the MIB Field Options Template or export a separate Options Template and a Data Record that maps MIB OIDs in mibObjectIdentifier fields to the object details.

If only a few fields are being exported, then including extra type data in the MIB Field Options export will be more efficient.

The MIB Field Options Template for a non-columnar field with extra MIB object details is shown in Figure 10.

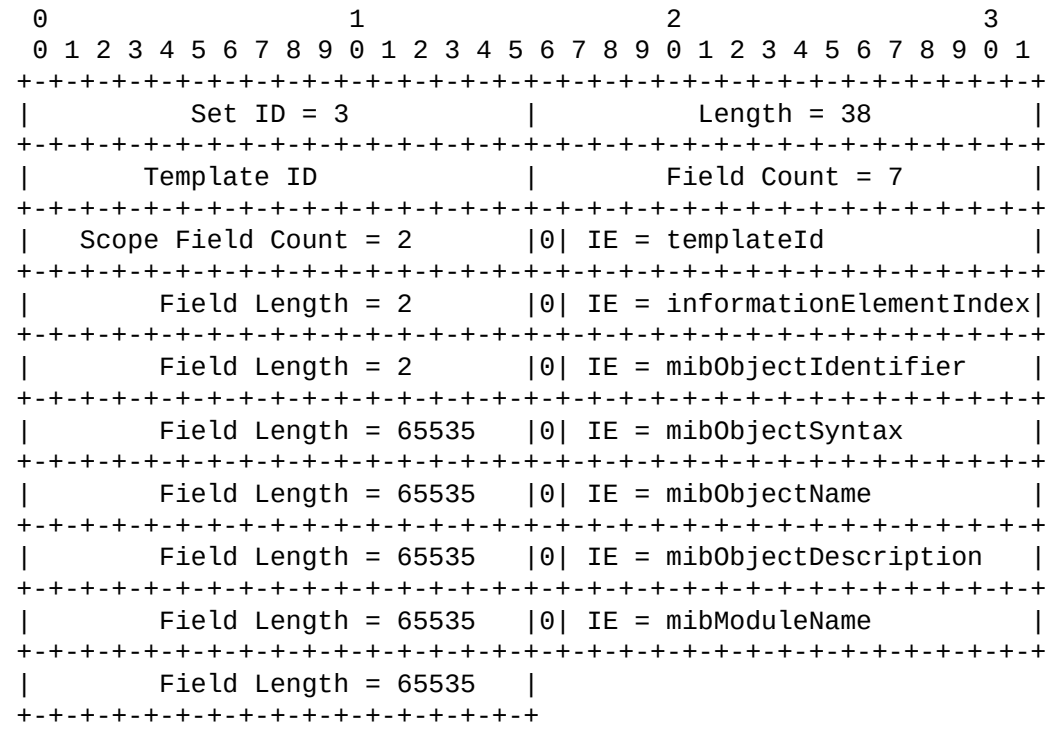


Figure 10: MIB Field Options Template for a Non-columnar Field with Extra MIB Object Details

Where:

mibObjectSyntax

The MIB object syntax string as defined in Section 11.2.3.3.

Note that a separate mibObjectSyntax Information Element is required (rather than extend the existing "IPFIX Information Elements" subregistry [IANA-IPFIX] that contains informationElementDataType) because the SYNTAX clause could contain almost any name.

mibObjectName

The textual name of a mibObjectIdentifier object.

mibObjectDescription

The textual description for a mibObjectIdentifier.

mibModuleName

The textual name of the MIB module that defines a MIB object.

Note the use of variable-length encoding for the mibObjectIdentifier, mibObjectSyntax, mibObjectName, mibObjectDescription, and mibModuleName, since these are all string fields.

The MIB details can be exported in Data Records specified using a regular IPFIX Options Template Record [RFC7011], as shown in Figure 11. This may be more efficient, as the bulk of this data is text based and SHOULD be exported only once to the Collecting Process if there are many MIB objects being exported. This prevents this large textual data from being included for every use of a mibObjectValue Information Element.

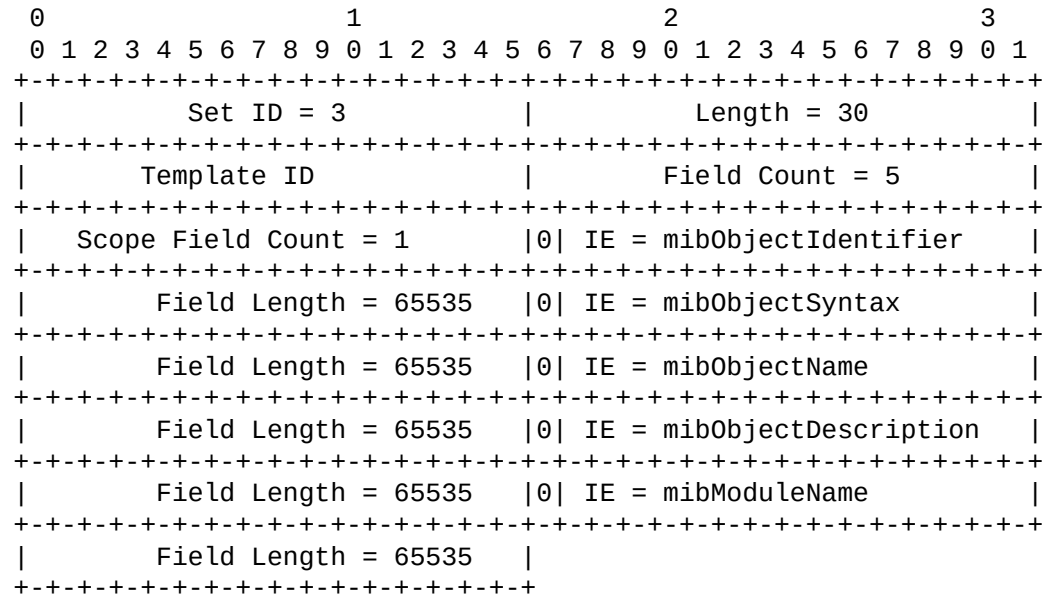


Figure 11: Alternative mibObjectIdentifier Options Template Set with Object Details

5.5. Use of Field Order in the MIB Field Options Template

The MIB Field Options Template export makes use of the informationElementIndex [IANA-IPFIX] to specify which field in the Template the metadata relates to; this avoids any ordering constraints on the Data Template. The mibObjectValue Information Elements in an IPFIX export can be in any order in the export packet. However, fields used as an INDEX MUST be in the same order as the order specified in the INDEX clause of the conceptual row MIB object.

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 MIB Field Options Data MUST be resent to reflect the new ordering. A new Template ID MUST be used to reflect that the ordering has changed. Older MIB Field Options Data may refer to the incorrect field.

A templateId [IANA-IPFIX] is only locally unique within a combination of an Observation Domain and Transport Session. As such, each MIB Field Options Data Record can only refer to templateIds within the same Observation Domain and session.

5.6. Identifying the SNMP Context

Each MIB OID is looked up in a specific context, usually the default context. If exporting a MIB OID value that isn't in the default context, then the context MUST be identified by including the mibContextEngineID (see Section 11.2.2.5) and mibContextName (see Section 11.2.2.6) fields in the MIB Field Options Template and associated MIB Field Options Data Records, or be included in the same Template as the mibObjectValue field.

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

The context information MAY be exported as part of the Template that includes the mibObjectValue Information Element, or the context information MAY be exported in the MIB Field Options Data Record that refers to the field. Context fields exported in the same Template MUST take precedence over those that refer to the Template. Context fields MUST apply to all mibObjectValue Information Elements in the same Template, and there MUST NOT be duplicates of mibContextName or mibContextEngineID in a Template.

So, a MIB Field Options Template MAY specify no context information, just the context engine ID or both the context engine and context name. This allows the Exporter to export the bulk of data in the default context and only tag those items that are required.

Since the MIB Field Options Template applies for all the Data Records of a Template, using context fields in the MIB Field Options Data Template requires that each mibContextEngineID / mibContextName pair have its own Template.

5.7. Template Management

Templates are managed as per Section 8 of [RFC7011], with the additional constraint that the MIB Field Options Template and MIB Field Options Data Records MUST be exported in the same IPFIX Message as any (Options) Template Record that uses a mibObjectValue Information Element.

When exporting over a Stream Control Transmission Protocol (SCTP) transport [RFC4960], the MIB Field Options Data Records MUST be exported reliably and in the same SCTP stream as their associated Templates per [RFC6526].

If a Template using a mibObjectValue Information Element is resent for any reason, the Data Records it depends on MUST be sent as well.

If a Template is replaced with a new (Options) Template, then a new MIB Field Options Data Record MUST be sent with the replacement referencing the new Template ID.

An Exporting Process SHOULD reuse MIB Field Options Template IDs when the Templates are identical. Each (Options) Template Record MUST still be accompanied by a copy of the MIB Field Options Template.

5.7.1. Large Messages

The requirement to export the MIB Field Options Template and MIB Field Options Data Records in the same IPFIX Message as any (Options) Template Record that uses a mibObjectValue Information Element may result in very large IPFIX Messages.

In environments with restricted Message sizes, and only when a reliable SCTP transport is being used, the MIB Field Options Template, MIB Field Options Data, Data Template, and Data Records MAY be exported in separate Messages in the same SCTP stream, provided that their order is maintained.

5.7.2. Template Withdrawal and Reuse

Data Records containing mibObjectValue Information Elements MUST NOT be exported if their corresponding Data Template or MIB Field Options Template has been withdrawn, since the MIB Field Options Template MUST be exported in the same IPFIX Message as the Data Template that it annotates, except as allowed by the caveat mentioned in Section 5.7.1.

MIB Field Options Template IDs MUST NOT be reused while they are required by any existing Data Templates.

5.8. Exporting Conceptual Rows and Tables

There are three approaches for an IPFIX Exporting Process to export the values of columnar objects:

1. Ignoring the indexing of columnar objects

2. Exporting conceptual rows / table objects using IPFIX Structured Data [RFC6313]

3. Exporting individual indexed columnar objects

Firstly, a subordinate columnar object may be used purely as a data type. In this case, there is no index information or relation to a conceptual row object provided by the Exporting Process.

Secondly, mibObjectValueRow or mibObjectValueTable can be used to export partial or complete conceptual rows, using IPFIX Structured Data [RFC6313].

Thirdly, in a mixed option/data IPFIX/MIB Template, the mibObjectValue Information Element can have the values of the INDEX clause of the conceptual row provided by other fields in the Data Record. In this case, each mibObjectValue Information Element must specify which other field(s) in the Template is providing the index information.

5.8.1. Exporting Conceptual Rows - Indexing

This document defines two forms of indexing that can be used for conceptual row MIB objects:

Indexing based on IPFIX Structured Data [RFC6313] is used solely by the mibObjectValueRow Information Element. Each conceptual row of the MIB object corresponds to a single Data Record exported. The index fields defined in the INDEX clause of the MIB object MUST all be present in the same order as the Scope Fields. This allows a simple table export of a conceptual row MIB object without any extra fields required to indicate which fields make up the conceptual row INDEX.

Field-based indexing is used by giving each mibObjectValue Information Element a mibIndexIndicator to flag the required index fields. This allows complex indexing or mixing of existing IPFIX Information Elements with MIB fields, with minimum overhead. It also allows multiple columnar MIB objects from different conceptual rows to be exported with complete indexing in one IPFIX Template.

5.8.2. Exporting Conceptual Rows - mibObjectValueRow

The simplest approach to exporting a complete or partial conceptual row object is done with the mibObjectValueRow Information Element.

This is an IPFIX Structured Data subTemplateList Information Element as detailed in [RFC6313]. The Template specified MUST be an Options Template. It also MUST have the fields specified in the INDEX clause of the conceptual row object as the Scope Fields in the MIB Field Options Template and Data Set.

An overview of this architecture is given in Figure 12. This shows that the full MIB object type definition OID is exported for the mibObjectValueRow conceptual row field but that the individual columnar objects only require the sub-identifier to be exported. To make the diagram clearer, the Templates for the MIB Field Options Templates are not shown.

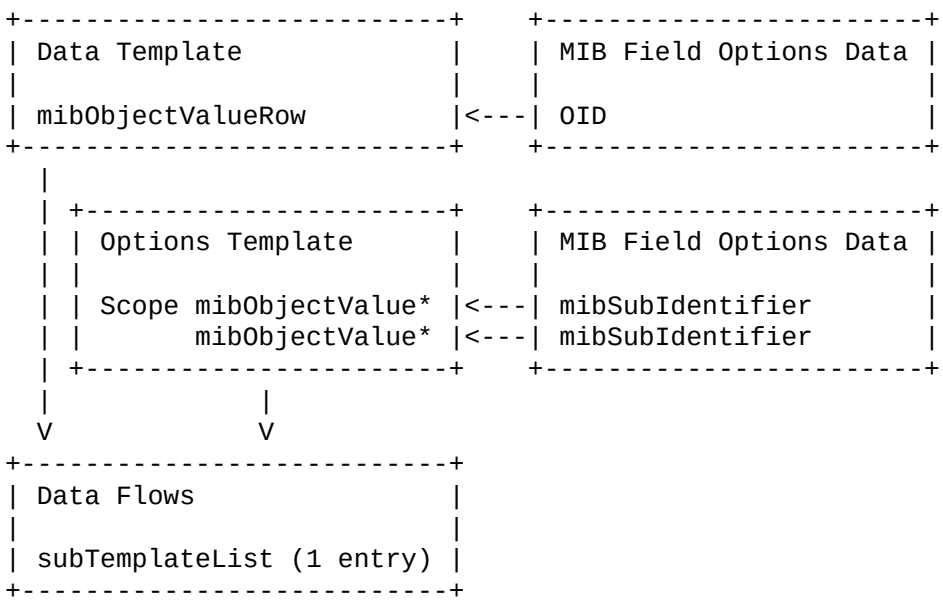


Figure 12: Architecture for Exporting Conceptual Rows with mibObjectValueRow

The mibIndexIndicator is not required for each individual mibObjectValue Information Element, as mibObjectValueRow provides a structure that includes the index details.

When indexing based on IPFIX Structured Data [RFC6313] is used, all Scope Fields MUST be the INDEX objects in the same order as defined in the INDEX clause of the conceptual row being exported.

Each conceptual table MIB object has two related OIDs. There is an OID that refers to the table with the syntax of SEQUENCE OF and an OID that refers to each entry or conceptual row with the syntax of SEQUENCE. The OID for the SEQUENCE of a conceptual row MUST be exported.

For example, in the IF-MIB [RFC2863], the OID for ifEntry should be exported rather than the OID for ifTable. The OID for the table (in this case, ifTable) can be derived by removing one sub-identifier from the ifEntry OID.

The full OID for the conceptual row MIB object type definition being exported with the mibObjectValueRow Information Element MUST be exported. However, the fields that are members of the conceptual row need not have the full OID of their MIB object type definition exported. Instead, the mibSubIdentifier Information Element can be used to document which entry in the conceptual row the field is.

In this case, the exported Flow will contain a single complete or partial row from a table inside a single field of the Data Record.

There may be MIB objects that are specified in the INDEX of the conceptual row but not columnar objects of the same conceptual row; for these, the Exporter MUST provide the full OID in a mibObjectIdentifier field.

So, for a conceptual row object with the OID "1.2.3.4.5.6" the OID of the type definitions for columnar objects "1.2.3.4.5.6.1" "1.2.3.4.5.6.2" can be exported with just a mibSubIdentifier of "1" and "2", respectively.

The mibObjectValue Information Elements exported using the mibObjectValueRow export MUST all either be objects defined in the INDEX clause, columnar objects of the same conceptual row object, or columnar objects that augment the same conceptual row.

The IPFIX Structured Data [RFC6313] subTemplateList format requires the Structured Data Type Semantics to be specified. Unless there is a more appropriate option in the "IPFIX Structured Data Types Semantics" subregistry [IANA-IPFIX], the "undefined" Structured Data Type Semantics can be used.

Figure 13 shows an IPFIX Template for an IPFIX Structured Data [RFC6313] export of a conceptual row, while Figure 14 shows an IPFIX Options Template for a complete conceptual row with five columns and two index fields. Figure 15 shows the MIB Field Options Template for a conceptual row field. Figure 16 shows the MIB Field Options Template for the columns inside the conceptual row. Figure 17 shows the OID Data for the conceptual row that will be exported.

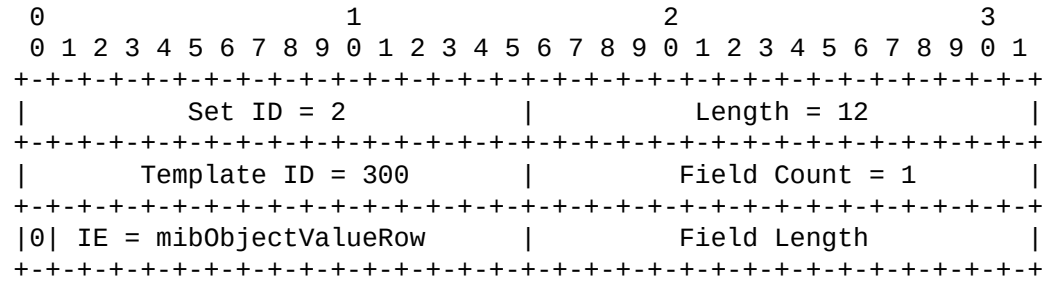


Figure 13: IPFIX Template for a Conceptual Row

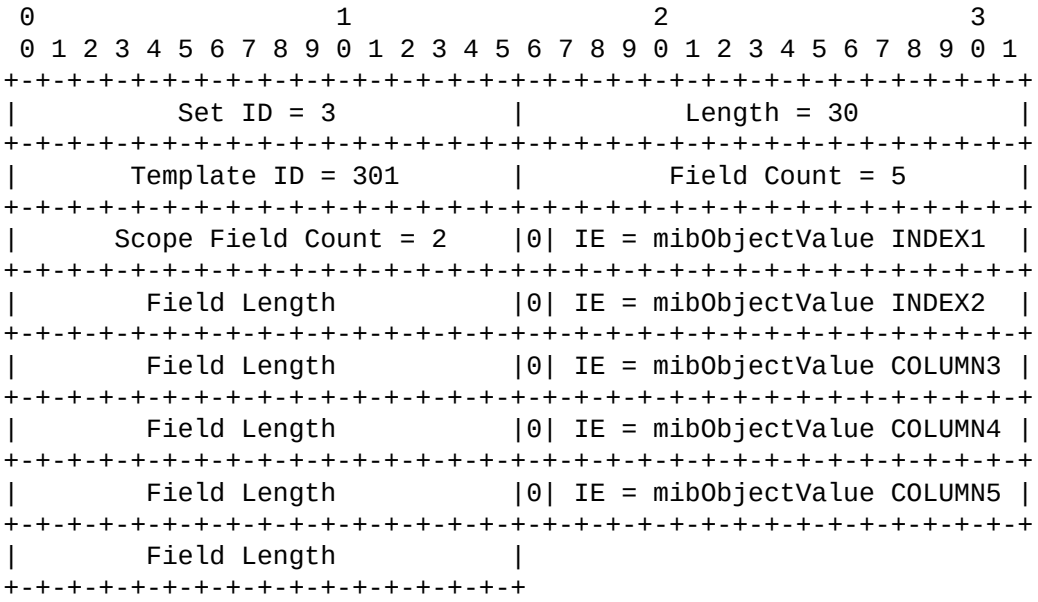


Figure 14: IPFIX Options Template for a mibObjectValueRow with Five Columns and Two Index Fields

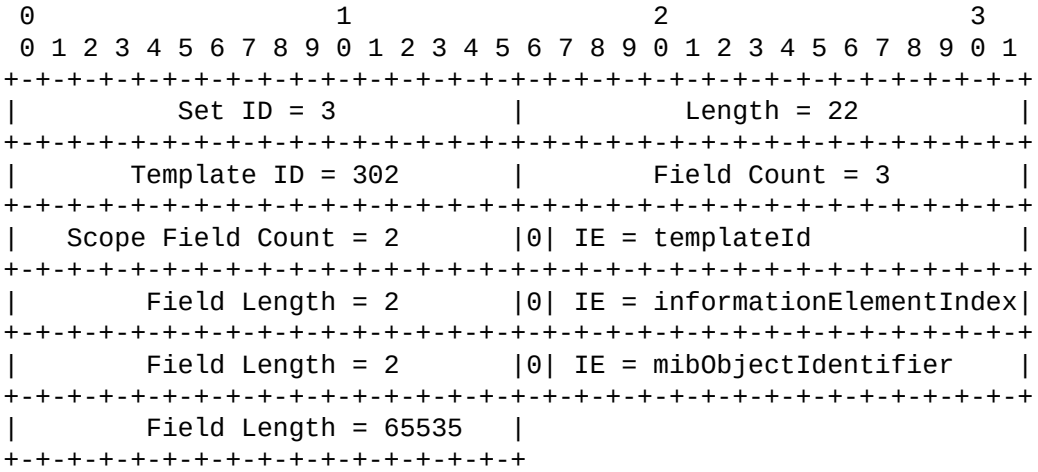


Figure 15: MIB Field Options Template for a Conceptual Row Object

Where:

templateId

The templateId for the MIB option that will be exported.

mibObjectIdentifier

The MIB OID for the conceptual row that is being exported. Note the use of variable-length encoding for this field.

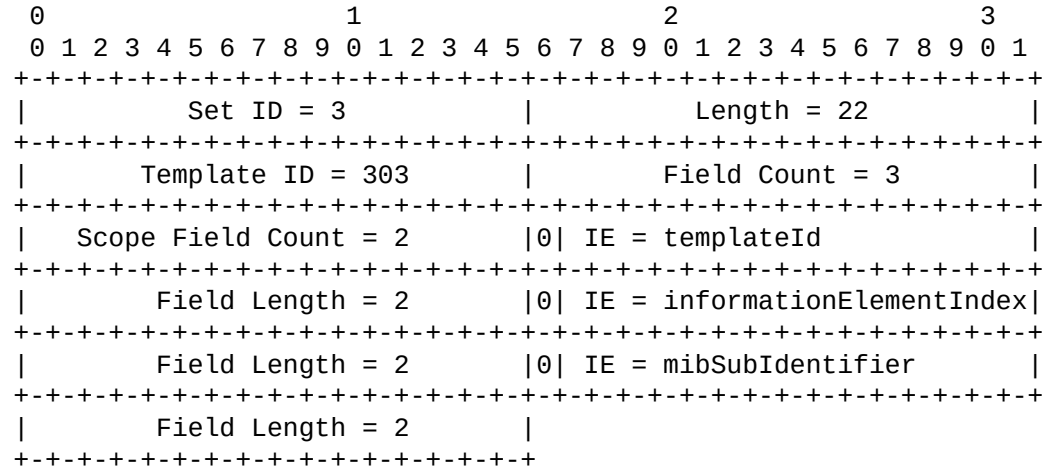


Figure 16: MIB Field Options Template for Columnar Objects of a Conceptual Table

Where:

templateId

The templateId used will be for the Template referred to in the subTemplateList of the mibObjectValueRow that will be exported.

mibSubIdentifier

The sub-identifier that specifies the columnar object's ID within the conceptual row.

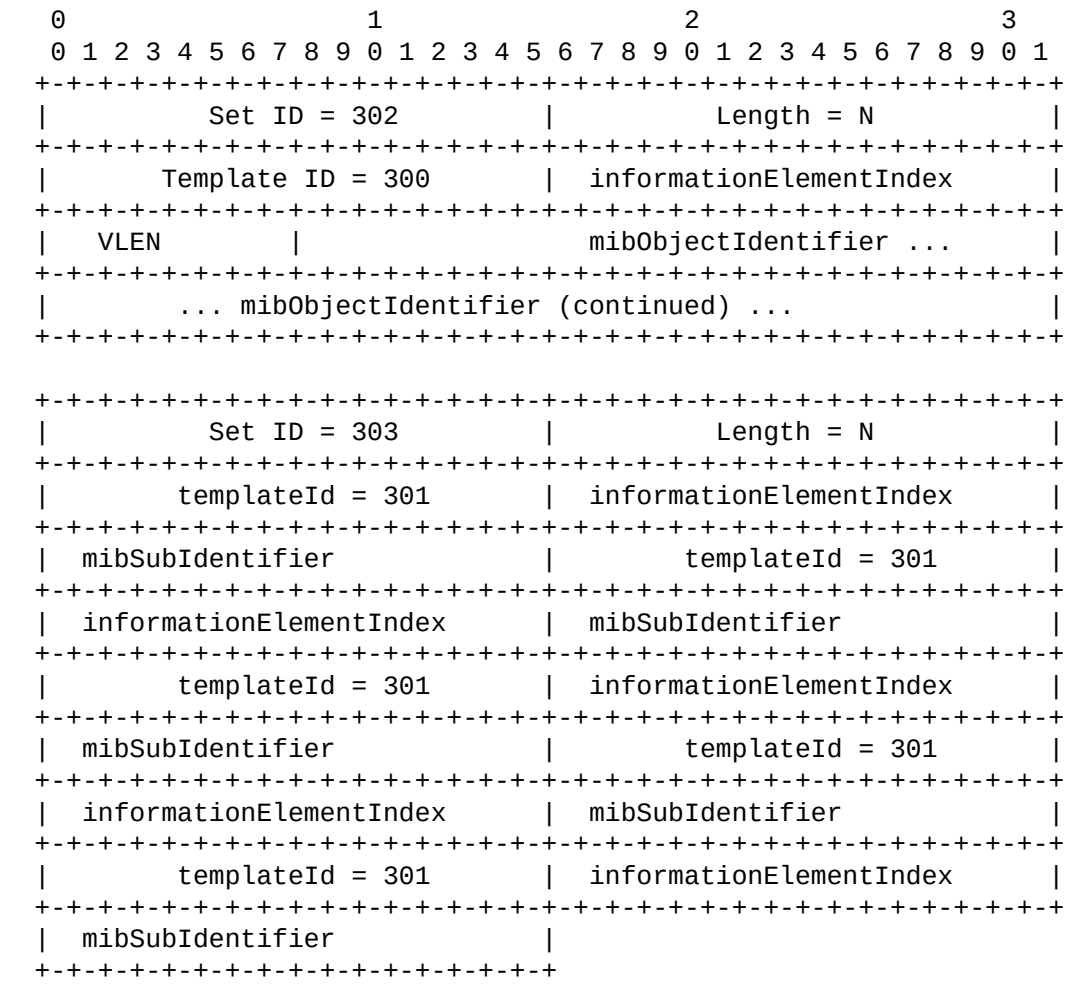


Figure 17: mibOption Data Record for the Conceptual Row

Where:

mibObjectIdentifier

Will contain the OID for the conceptual row as a whole.

mibSubIdentifier

The mibSubIdentifier fields will contain the extra sub-identifier that, when added to the OID for the conceptual row, gives the full OID for the object.

5.8.3. Exporting Conceptual Rows - AUGMENTS

SMIv2 defines conceptual rows as having either an INDEX clause or an AUGMENTS clause. Conceptual row definitions with an AUGMENTS clause extend an existing base conceptual row with an INDEX clause. It is not possible in SMIv2 to augment a conceptual row that itself has an AUGMENTS clause. The base table and the augmentation have an identical INDEX.

Since augmentations allow adding extra columns to existing tables, it is beneficial to be able to support them easily in IPFIX exports of conceptual rows.

The mibObjectValueRow OID MAY refer to either the base table with the INDEX clause or a conceptual row with an AUGMENTS clause. The mibSubIdentifier in any MIB Field Options Data Record MUST always refer to the OID exported for the mibObjectValueRow Information Element.

If the mibObjectValueRow OID refers to a base table, then any extra columns from conceptual rows with an AUGMENTS clause MUST have their full OID exported.

If the mibObjectValueRow OID refers to a conceptual row that augments another conceptual row using the AUGMENTS clause, then any MIB fields from the original table's INDEX or columnar objects MUST NOT use the mibSubIdentifier and MUST instead export the full OID in a mibObjectIdentifier.

If the mibObjectValueRow refers to an augmenting conceptual row, the Scope Fields of the Template used in the subTemplateList MUST have the index fields from the base table, in the same order as its scope. This is identical to the Scope Field requirements for conceptual rows with an INDEX clause.

This flexibility is provided so that the conceptual rows with the most columns can be exported using the more efficient mibSubIdentifier. For example, exporting a complete set of augmentation columns would only require the full OIDs for the MIB objects in the INDEX.

It is possible to export MIB object columns from multiple augmenting conceptual rows. If this is done, then the base table SHOULD be used as the main OID for the mibObjectValueRow.

5.8.4. Exporting Conceptual Tables - mibObjectValueTable

Multiple rows of a conceptual table can be exported in the mibObjectValueTable Information Element (Section 11.2.1.10). This allows a set of conceptual rows corresponding to a conceptual table to be exported as a single field. Therefore, a complete set of rows can be exported as a single field with other Information Elements in a Template. In this fashion, several complete conceptual tables could be exported in one packet.

As also specified for mibObjectValueRow (Section 5.8.2), the more specific (i.e., full) OID of the SEQUENCE entity MUST be exported.

The format of mibObjectValueTable is identical to mibObjectValueRow, except that the length of the subTemplateList may be zero or more entries.

All the other, i.e., non-length, requirements for mibObjectValueRow in Section 5.8.2 apply to mibObjectValueTable.

An overview of this architecture is given in Figure 18. This architecture is similar to the architecture shown in Figure 12.

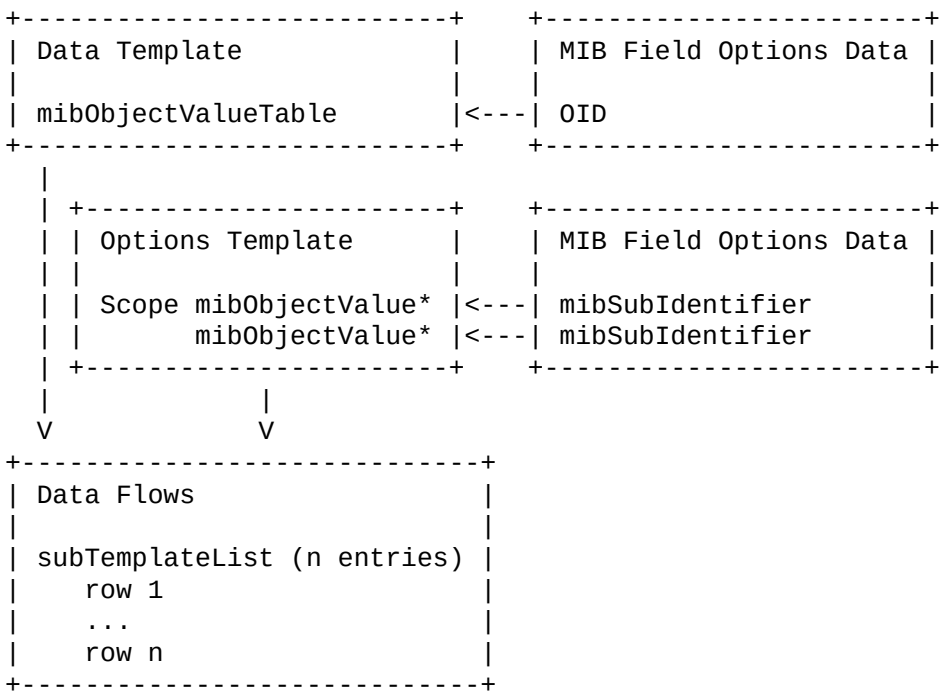


Figure 18: Architecture for Exporting Conceptual Tables with mibObjectValueTable

5.8.5. Exporting Columnar Objects: Using mibIndexIndicator

The other option for indexing a columnar object that is part of a conceptual table is explicit indexing. In this case, the Options Template Set scope may contain either non-index fields or columnar MIB objects from multiple conceptual rows being exported. In this case, each mibObjectValue Information Element requires the mibIndexIndicator with the bits set for the fields that are used to index that individual columnar object.

The index fields MUST be in the "correct" order as defined in the conceptual row that each columnar object is a member of.

If a mibObjectValue Information Element that is being indexed using mibIndexIndicator is being used as an Options Template Scope Field, then all fields used to index that field MUST also be Scope Fields.

Figure 19 shows the MIB Field Options Template for an indexed MIB columnar object.

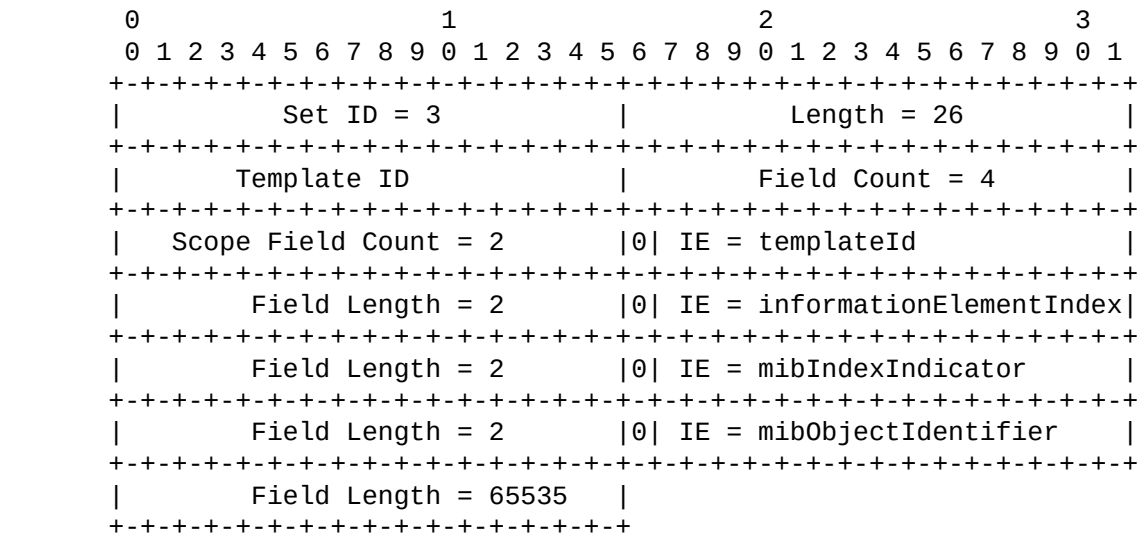


Figure 19: MIB Field Options Template for an Indexed MIB Columnar Object

Where:

mibIndexIndicator

The MIB Index Indicator IPFIX Information Element that marks which fields in the Data 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 OID for the MIB object type definition exported in mibObjectIdentifier to get the OID for the instance of the MIB object.

Elements used to index MIB objects MUST be exported in the same order as they are specified in the index field of the conceptual table they belong to.

mibObjectIdentifier

Note the use of variable-length encoding for this field.

6. Example Use Cases

6.1. Non-columnar 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 provide details for the following two Information Elements:

1. flowStartSeconds from [IANA-IPFIX], 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 20 shows the exported Template Set detailing the Template Record for exporting the number of established TCP connections.

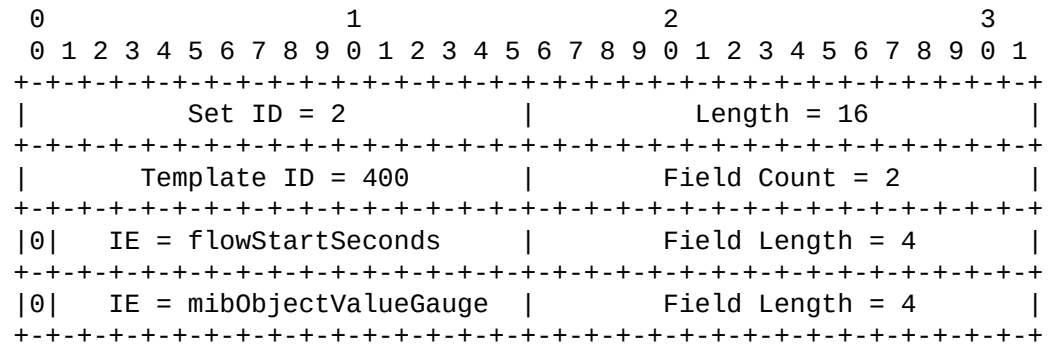


Figure 20: Example of tcpCurrEstab Template Set

Figure 21 shows the exported MIB Field Options Template Set detailing the metadata that will be exported about the mibObjectValueGauge Information Element in Template 400 in Template Record.

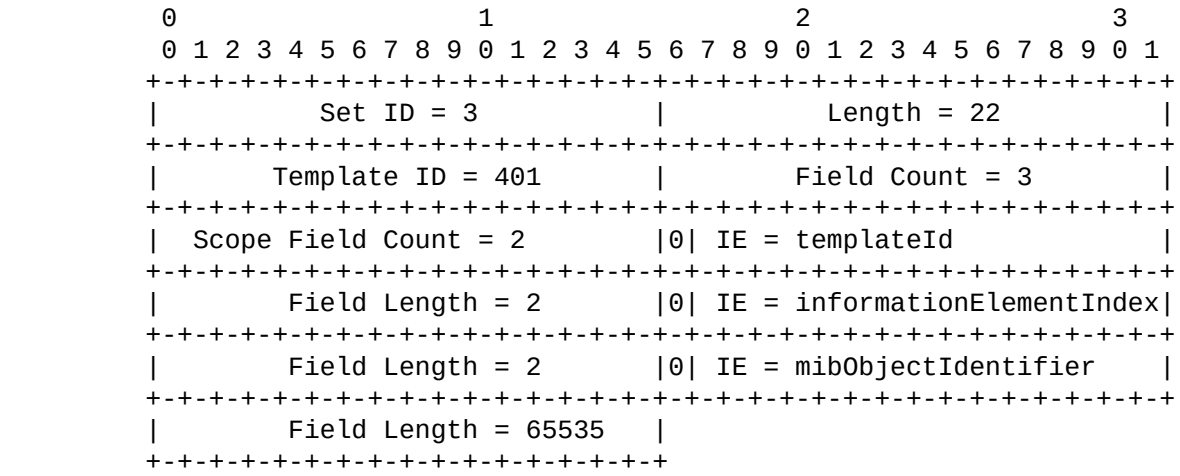


Figure 21: Example of tcpCurrEstab MIB Field Options Template Set

Figure 22 shows the exported MIB Field Options Data Set detailing the metadata that will be exported about the mibObjectValueGauge Information Element in Template 400 in Template Record.

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 [X.690] as "06072B0601020106 09" in the Data Record, which takes 9 octets.

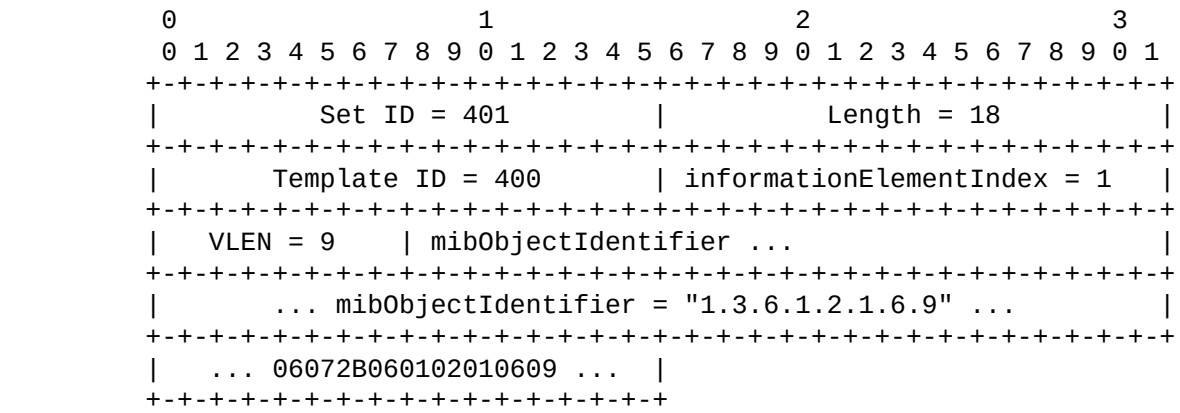


Figure 22: Example of tcpCurrEstab MIB Field Options Data Set

Figure 23 shows the start of the Data Set for exporting the number of established TCP connections (see Section 6.1).

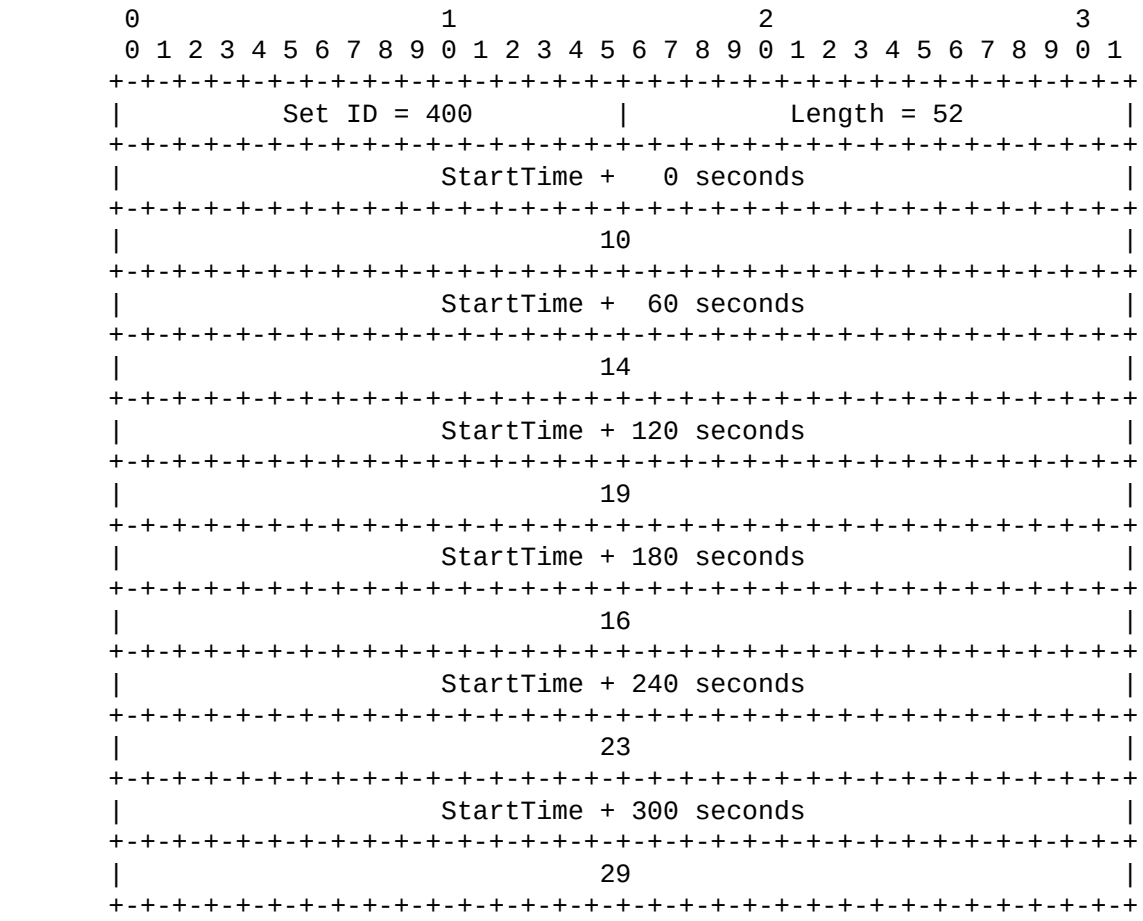


Figure 23: Example of tcpCurrEstab Data Set

6.2. Enterprise-Specific MIB Object: Detailing CPU Load History

For the sake of demonstration, an enterprise-specific MIB object from the CISCO-PROCESS-MIB [CISCO-PROCESS-MIB] is chosen. This example would be valid with any enterprise-specific MIB module.

The CPU usage of a remote network device with one CPU 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.

Although the cpmCPUTotal1minRev MIB object is a columnar object in a conceptual row, if there is only one CPU no extra information is conveyed by providing the index field. So, in this case, it is acceptable to not export the cpmCPUTotalIndex MIB object. If there were multiple CPUs, it would be appropriate to include the cpmCPUTotalIndex field and specify the relationship.

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 provide details for the following two Information Elements:

1. flowStartSeconds from [IANA-IPFIX], 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 1-minute period.

Figure 24 shows the exported Template Set detailing the Template Record for exporting CPU Load (see Section 6.2).

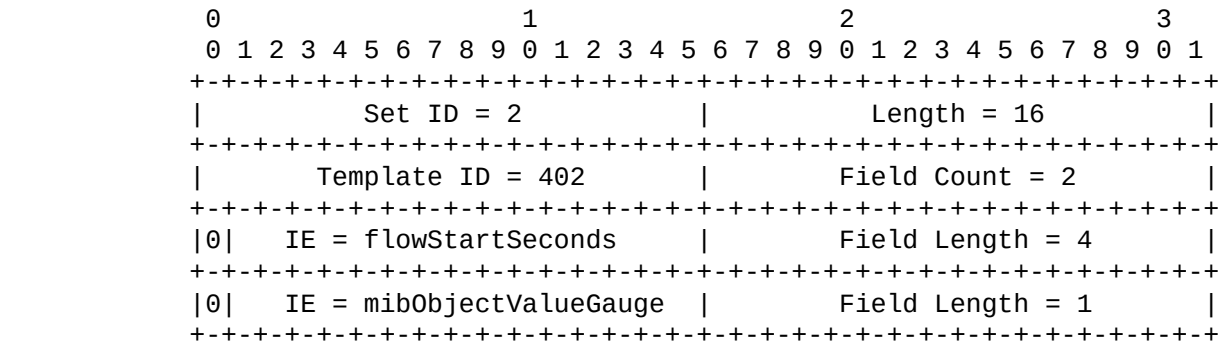


Figure 24: Example of CPU Load Template Set

Figure 25 shows the exported Template Set detailing the MIB Field Options Template for exporting CPU Load (see Section 6.2). Note: This is identical to the MIB Field Options Template given in Figure 21, so the same Template could have been reused.

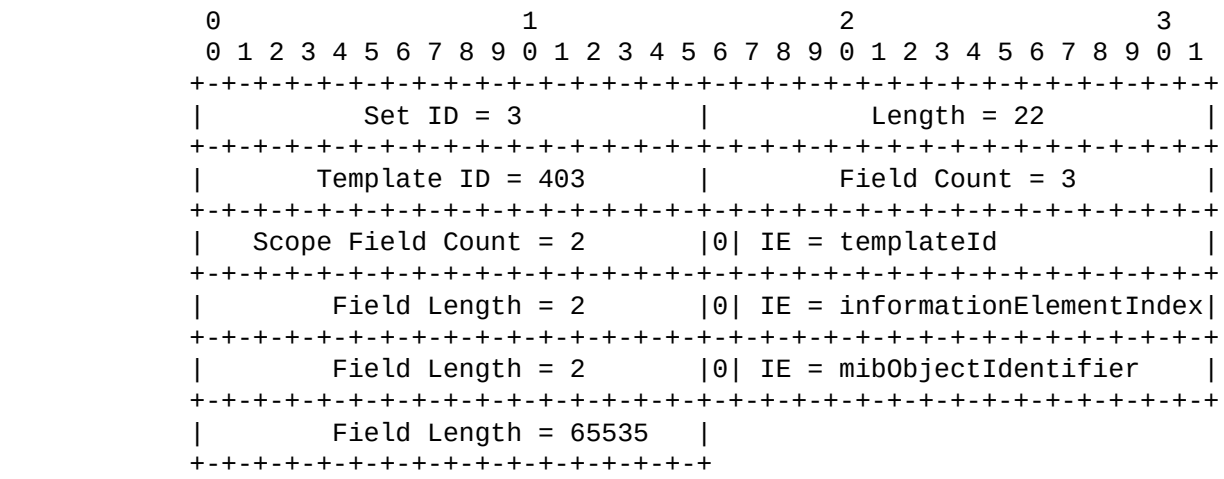


Figure 25: Example of CPU Load MIB Field Options Template Set

Figure 26 shows the exported MIB Field Options Data Set detailing the metadata that will be exported about the mibObjectValueGauge Information Element in Template 402 in Template Record (see Section 6.2).

The OID for the cpmCPUTotal1minRev has been encoded using ASN.1/BER to "060D2B0601040109096D0101010107" at 15 octets long.

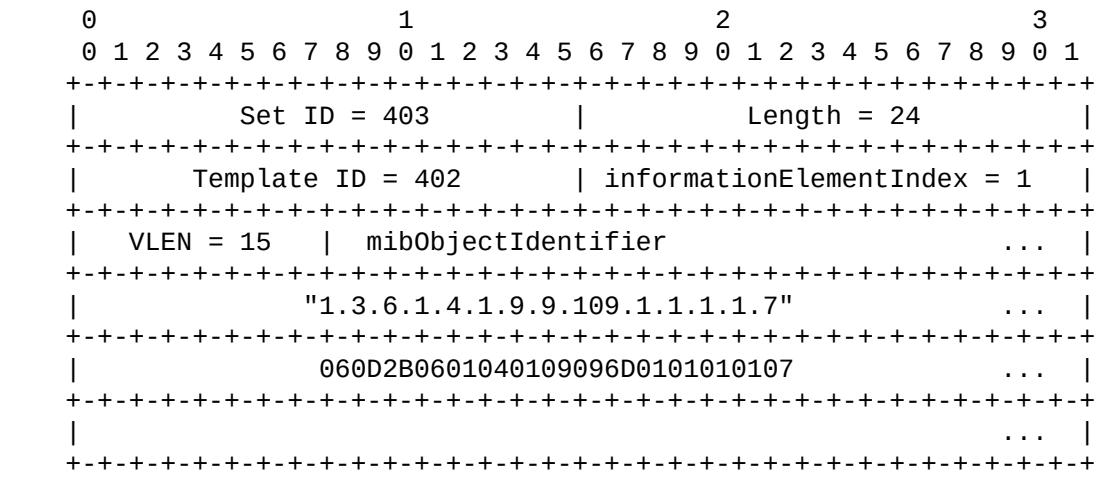


Figure 26: Example of CPU Load MIB Field Options Data Set

Note that although cpmCPUTotal1minRev is 32 bits long, reduced-size encoding [RFC7011] has been used to encode it within a single octet. The encoding size was specified by setting the length for the mibObjectValueGauge field to 1 octet in the main Data Template; see Figure 24.

This example stresses that, even though the OID cpmCPUTotal1minRev is enterprise-specific, the E bit for the mibObjectValueGauge and mibObjectIdentifier is set to 0, because the mibObjectValueGauge and mibObjectIdentifier Information Elements are 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 Conceptual Row: The OSPF Neighbor Row

Many conceptual tables are already defined in standard and proprietary MIBs. These can be exported with a minimum of overhead by using the mibObjectValueRow. This allows the Exporting Process to unambiguously define the INDEX for the entire conceptual row as the Scope Fields of an Options Template Set. The use of a MIB Field Options Template with mibSubIdentifier being used means that each individual columnar object does not need to have its OID exported to the Collector.

The ospfNbrTable, defined in the OSPF MIB [RFC4750], consists of ospfNbrEntry, which has the OID "1.3.6.1.2.1.14.10.1". Each mibObjectValueRow Data Record will therefore correspond to an ospfNbrEntry.

The following fields will be exported:

Object	ID	mibObjectValue	Len
ospfNbrIpAddr	ospfNbrEntry 1	mibObjectValueIPAddress	4
ospfNbrAddress-LessIndex	ospfNbrEntry 2	mibObjectValueInteger	4
ospfNbrRtrId	ospfNbrEntry 3	mibObjectValueIPAddress	4
ospfNbrState	ospfNbrEntry 6	mibObjectValueInteger	1

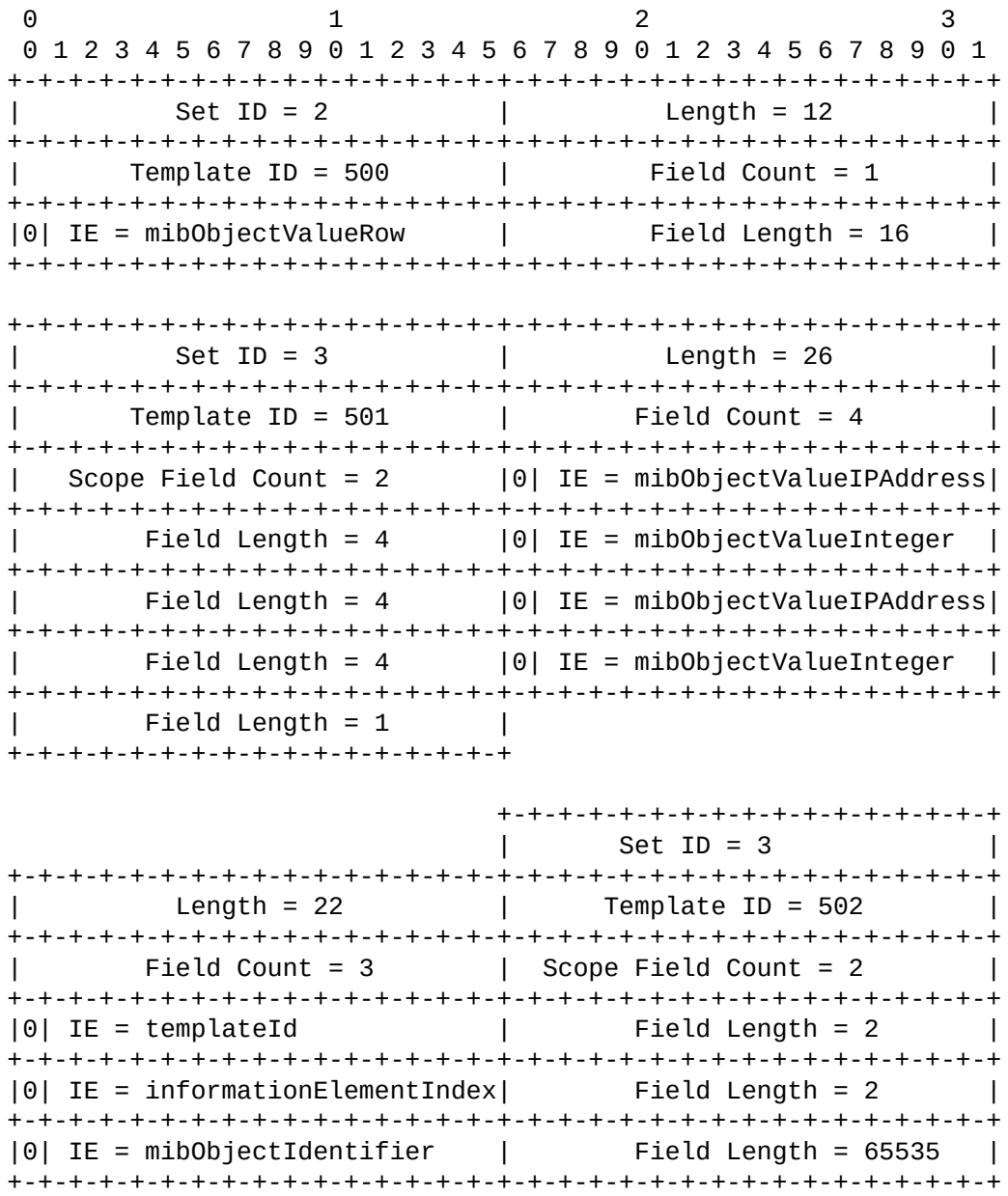
Table 4: OSPF Neighbor Entry Objects

The OIDs that will be used to export this table are shown in Table 5.

Entity	Full OID	Exported as
ospfNbrEntry	1.3.6.1.2.1.14.10.1	1.3.6.1.2.1.14.10.1
ospfNbrIpAddr	1.3.6.1.2.1.14.10.1.1	1
ospfNbrAddress-LessIndex	1.3.6.1.2.1.14.10.1.2	2
ospfNbrRtrId	1.3.6.1.2.1.14.10.1.3	3
ospfNbrState	1.3.6.1.2.1.14.10.1.6	6

Table 5: OSPF OIDs

Figure 27 shows the Templates exported to support the mibObjectValueRow. Figure 28 shows the example OID Data for the conceptual row exported in mibObjectValueRow. Figure 29 shows the example data export for a few neighbors in the table; Figure 29 also shows a Data Record formatted as per IPFIX Structured Data [RFC6313] and using the "undefined" (= 0xFF) semantic from the "IPFIX Structured Data Types Semantics" subregistry [IANA-IPFIX]. Note that the OID for ospfNbrEntry has been encoded using ASN.1/BER to "06082B060102010E0A01" at 10 octets long.


0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Set ID = 2
Length = 12
Template ID = 500
Field Count = 1
|0| IE = mibObjectValueRow
Field Length = 16
Set ID = 3
Length = 26
Template ID = 501
Field Count = 4
Scope Field Count = 2
|0| IE = mibObjectValueIPAddress|
Field Length = 4
|0| IE = mibObjectValueInteger
Field Length = 4
|0| IE = mibObjectValueIPAddress|
Field Length = 4
|0| IE = mibObjectValueInteger
Field Length = 1
Set ID = 3
Length = 22
Template ID = 502
Field Count = 3
Scope Field Count = 2
|0| IE = templateId
Field Length = 2
|0| IE = informationElementIndex|
Field Length = 2
|0| IE = mibObjectIdentifier
Field Length = 65535

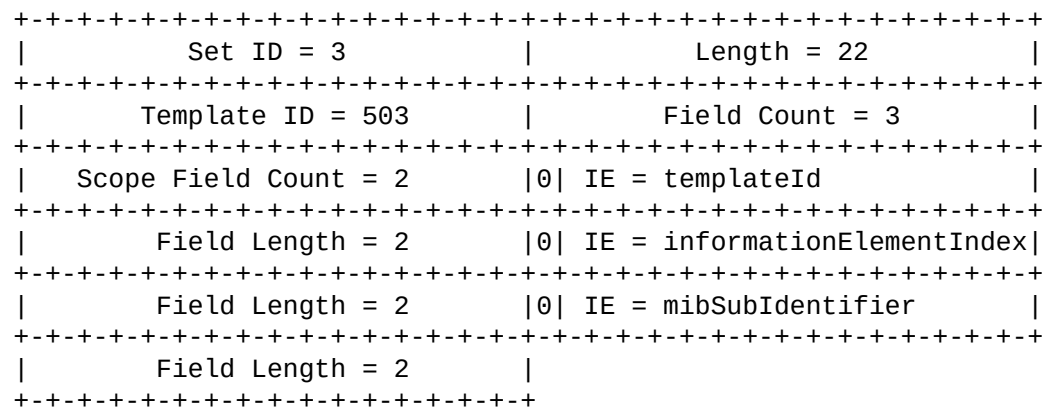


Figure 27: Example of ospfNbrEntry Template and Options Template Sets

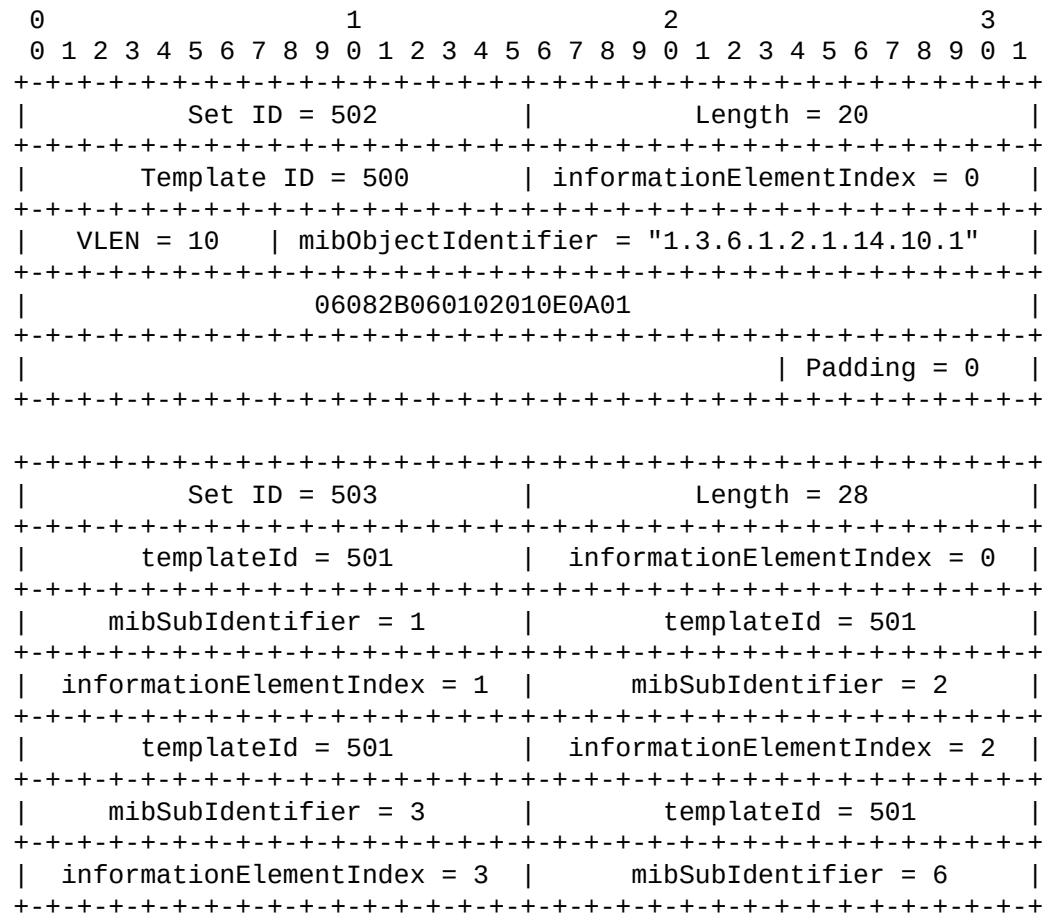


Figure 28: Example of ospfNbrEntry OID Data Export

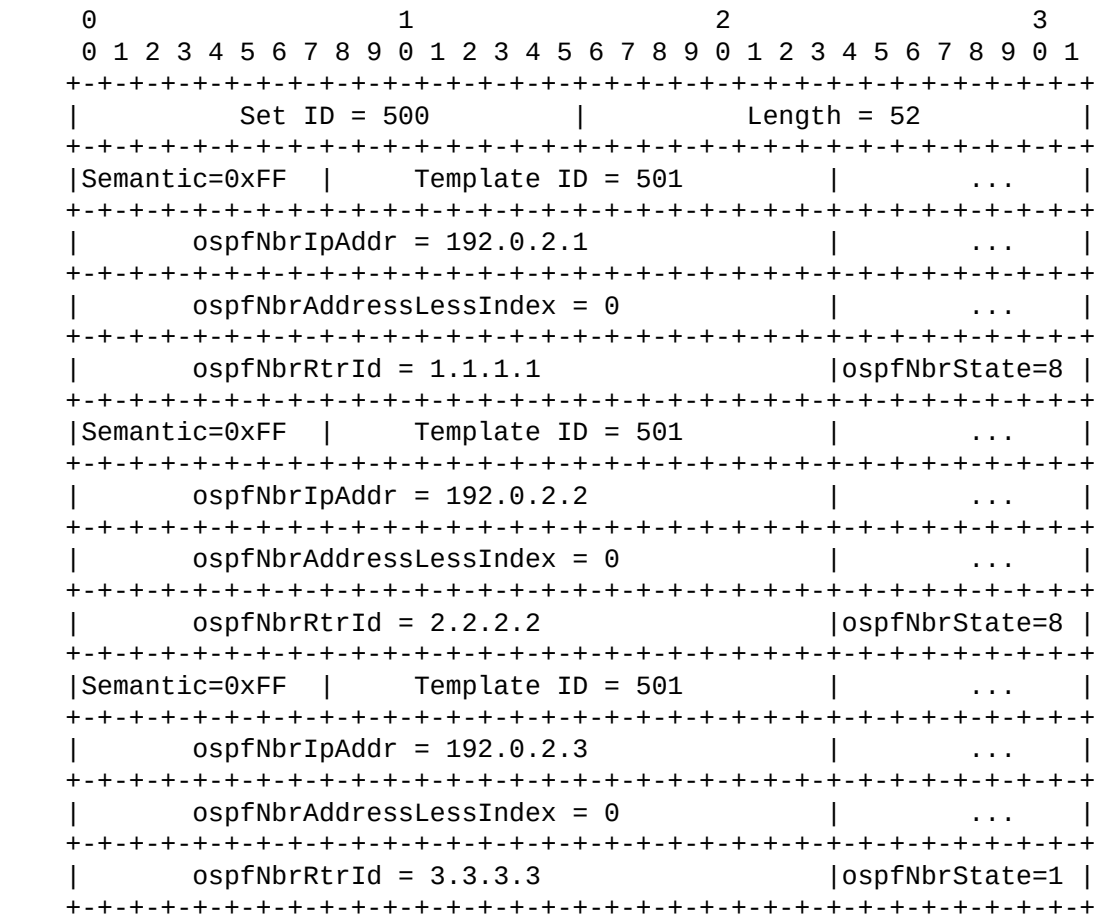


Figure 29: Example of Data Export for ospfNbrEntry

6.4. Exporting Augmented Conceptual Row: Mapping IF-MIB ID to Name

The ifTable, defined in the IF-MIB [RFC2863], is augmented by the ifXTable (defined in the same MIB module).

The OID of the ifEntry is 1.3.6.1.2.1.2.2.1, which is encoded using ASN.1/BER to "06082B06010201020201" at 10 octets long, while the OID of the augmenting ifXEntry is 1.3.6.1.2.1.31.1.1.1, which is encoded using ASN.1/BER to "060A2B060102011F01010101" at 12 octets long.

This example demonstrates how columnar objects from the base conceptual row and the augmenting row can be exported in a single mibObjectValueRow Information Element.

Table 6 shows the fields that will be exported.

ifIndex	ifType	ifMtu	ifName
1	ethernetCsmacd:6	1500	Ethernet 10
2	ethernetCsmacd:6	1500	Ethernet 20
3	ethernetCsmacd:6	1500	FastEthernet 30

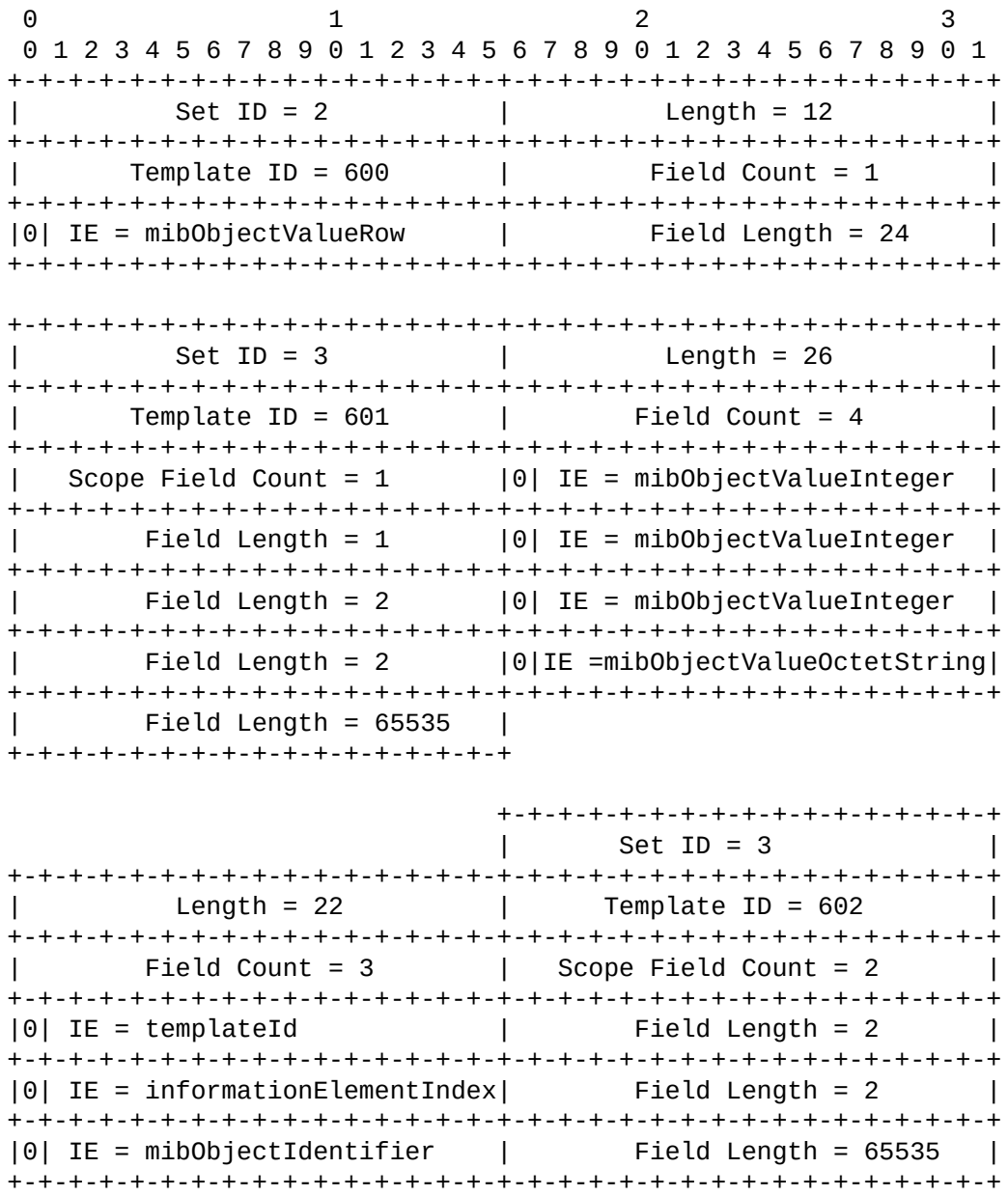
Table 6: IF-MIB Data

The OIDs that will be used to export this table are shown in Table 7.

Entity	Full OID	Exported as
ifEntry	1.3.6.1.2.1.2.2.1	OID = 1.3.6.1.2.1.2.2.1
ifIndex	1.3.6.1.2.1.2.2.1.1	subID = 1
ifType	1.3.6.1.2.1.2.2.1.3	subID = 3
ifMtu	1.3.6.1.2.1.2.2.1.4	subID = 4
ifName	1.3.6.1.2.1.31.1.1.1.1	OID = 1.3.6.1.2.1.31.1.1.1.1

Table 7: IF-MIB OIDs

Figure 30 shows the Templates exported to support the mibObjectValueRow Information Element. Figure 31 shows the example OID Data for the conceptual row exported in mibObjectValueRow to match Table 7. Figure 32 shows the example data export as per Table 6.


0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Set ID = 2
Length = 12
Template ID = 600
Field Count = 1
0| IE = mibObjectValueRow
Field Length = 24
Set ID = 3
Length = 26
Template ID = 601
Field Count = 4
Scope Field Count = 1
0| IE = mibObjectValueInteger
Field Length = 1
0| IE = mibObjectValueInteger
Field Length = 2
0| IE = mibObjectValueInteger
Field Length = 2
0|IE =mibObjectValueOctetString
Field Length = 65535
Set ID = 3
Length = 22
Template ID = 602
Field Count = 3
Scope Field Count = 2
0| IE = templateId
Field Length = 2
0| IE = informationElementIndex
Field Length = 2
0| IE = mibObjectIdentifier
Field Length = 65535

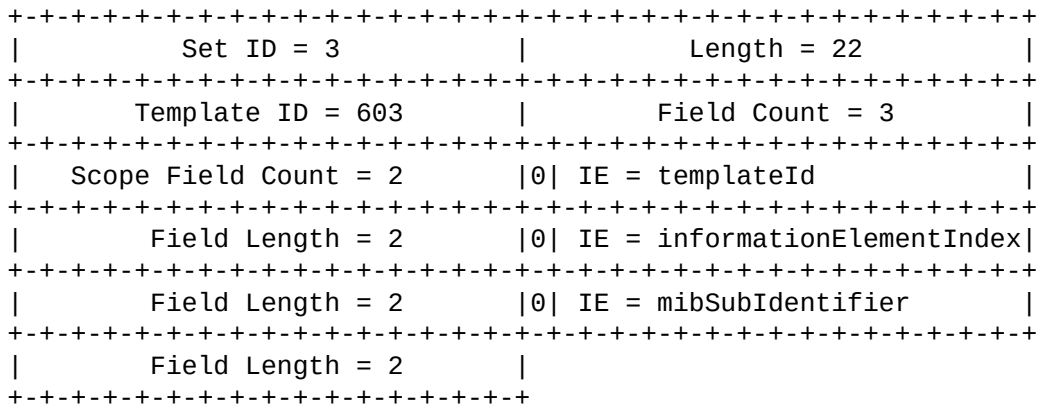


Figure 30: Example of Augmented ifEntry Template and Options Template Sets

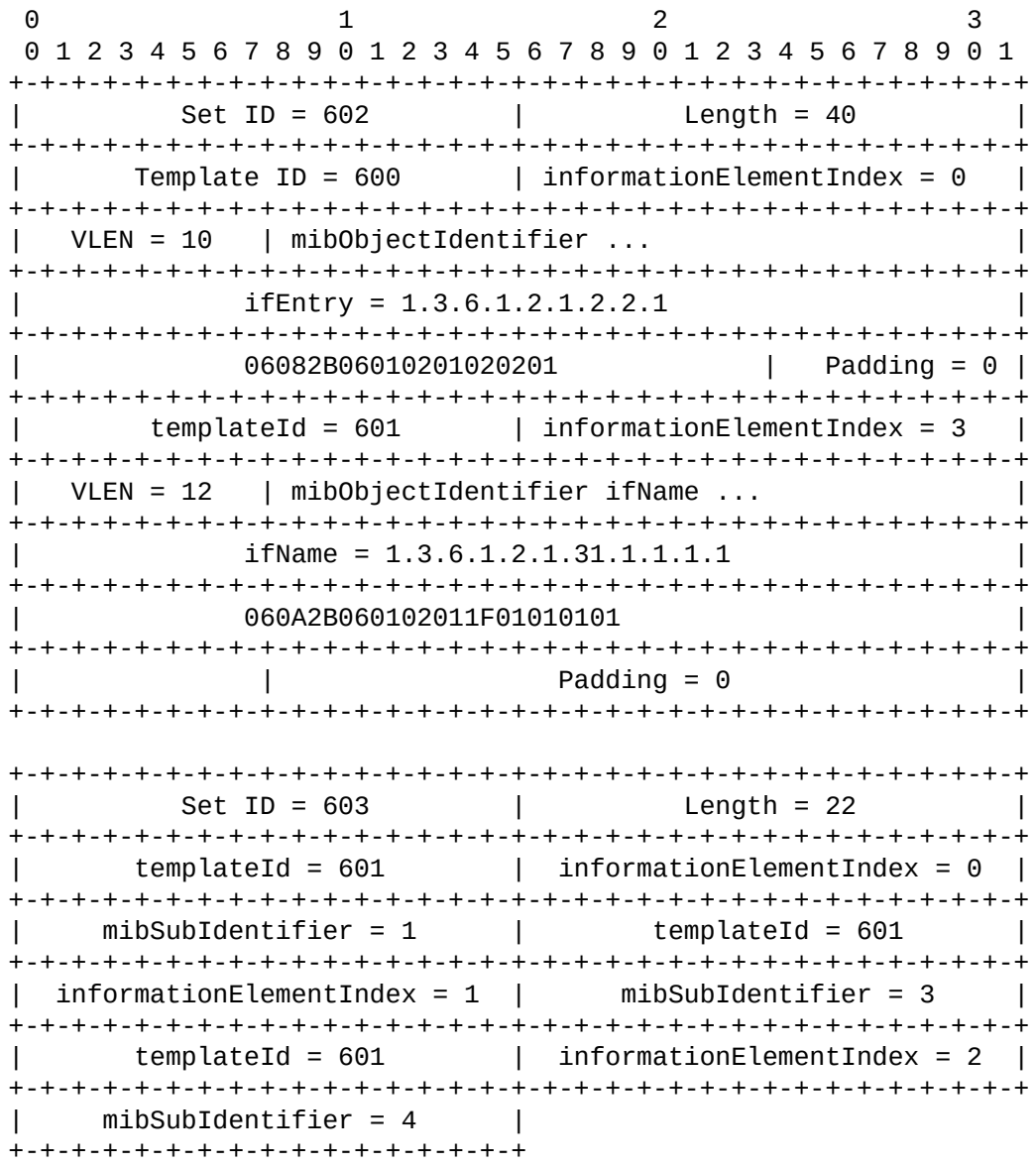


Figure 31: Example of Augmented ifEntry OID Data Export

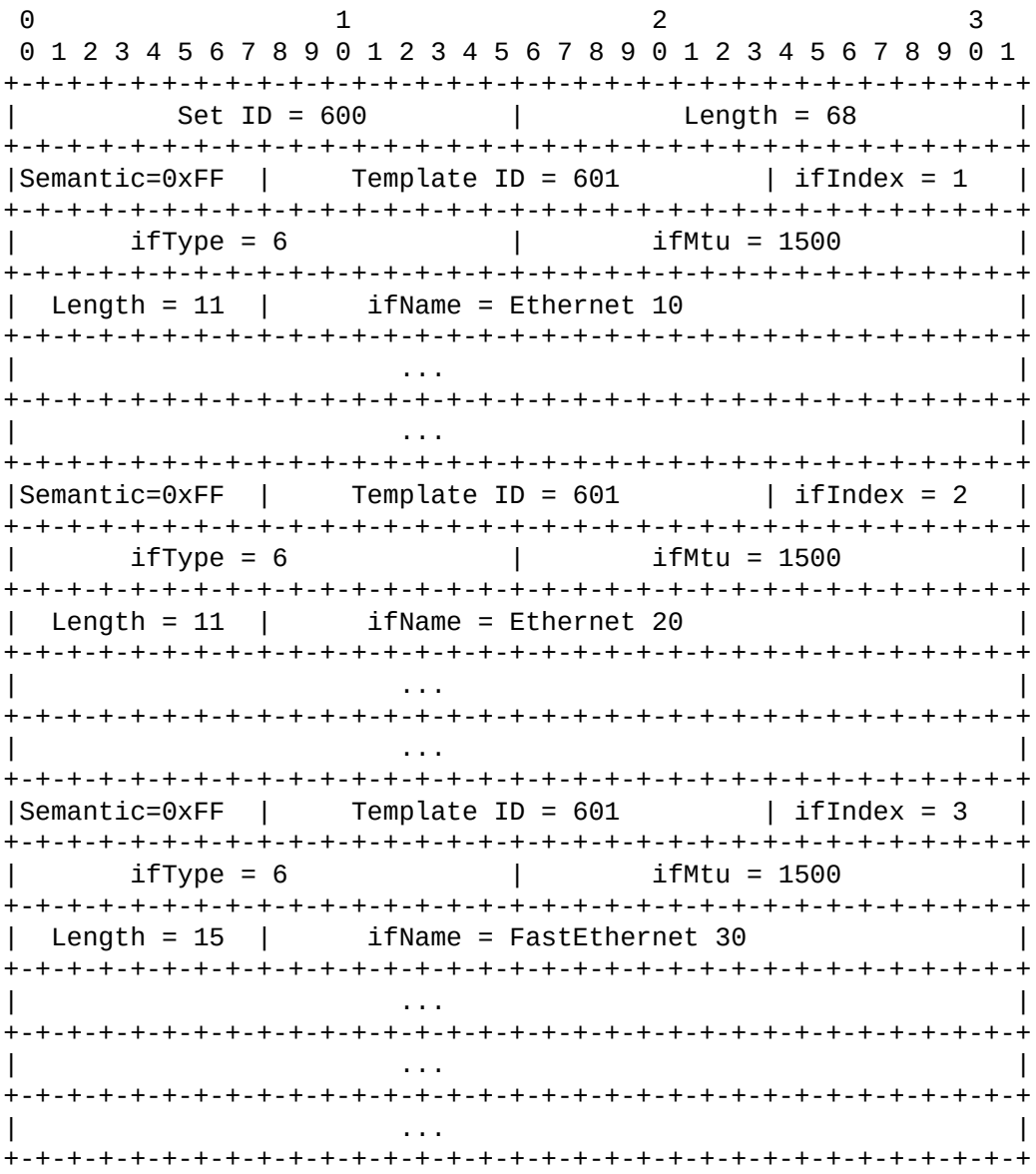


Figure 32: Example of Data Export for Augmented ifEntry

6.5. Exporting a Columnar Object: ipIfStatsInForwDatagrams

It may be that the full set of columnar objects that are supported by a conceptual row are not required to be exported. Rather than use the IPFIX Structured Data [RFC6313] method, the mibIndexIndicator method can be used to provide the relationship between fields.

This example shows the MIB objects that are part of the INDEX of the conceptual row being exported in the correct order and then being referred to by using mibIndexIndicator.

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

The ipIfStatsTable conceptual table is indexed by ipIfStatsIPVersion and ipIfStatsIfIndex.

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)
   (encoded using ASN.1/BER to "060A2B06010201041F030101" at
   12 octets long)

2. ipIfStatsIfIndex (1.3.6.1.2.1.4.31.3.1.2) (Scope Field)
   (encoded using ASN.1/BER to "060A2B06010201041F030102" at
   12 octets long)

3. ipIfStatsInForwDatagrams (1.3.6.1.2.1.4.31.3.1.12) (non-Scope
   Field)
   (encoded using ASN.1/BER to "060A2B06010201041F03010C" at
   12 octets long)

Note that ipIfStatsIfIndex has been reduced-size encoded to 2 octets in the following example. An exporting device with more interfaces would use the full length.

Figure 33 shows the exported Options Template Set.

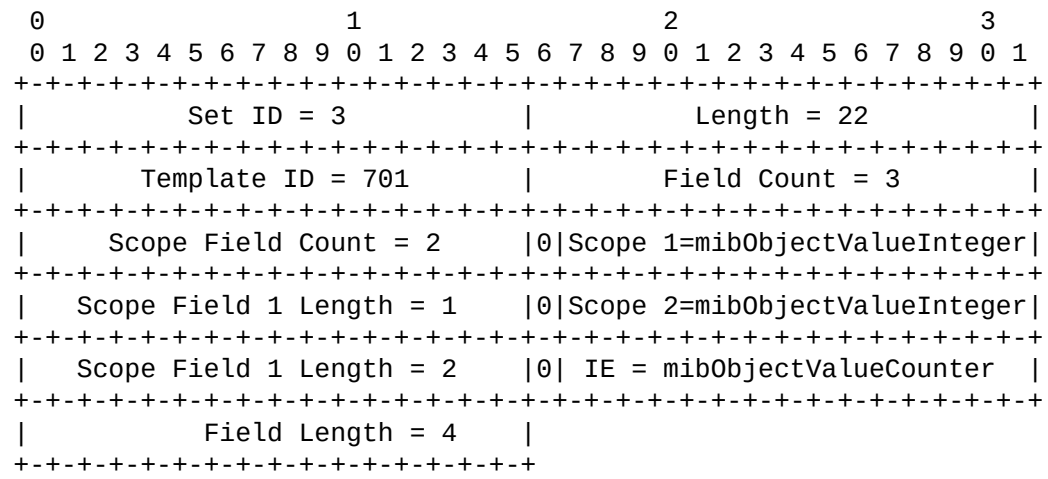


Figure 33: Example of an Options Template for an Indexed MIB Object with Two Index Objects

Figure 34 shows the exported MIB Field Options Template used to export the required mibObjectValue Information Element metadata. This example of the MIB Field Options Template includes the mibIndexIndicator to indicate that some of the other fields in the Data Records are index objects.

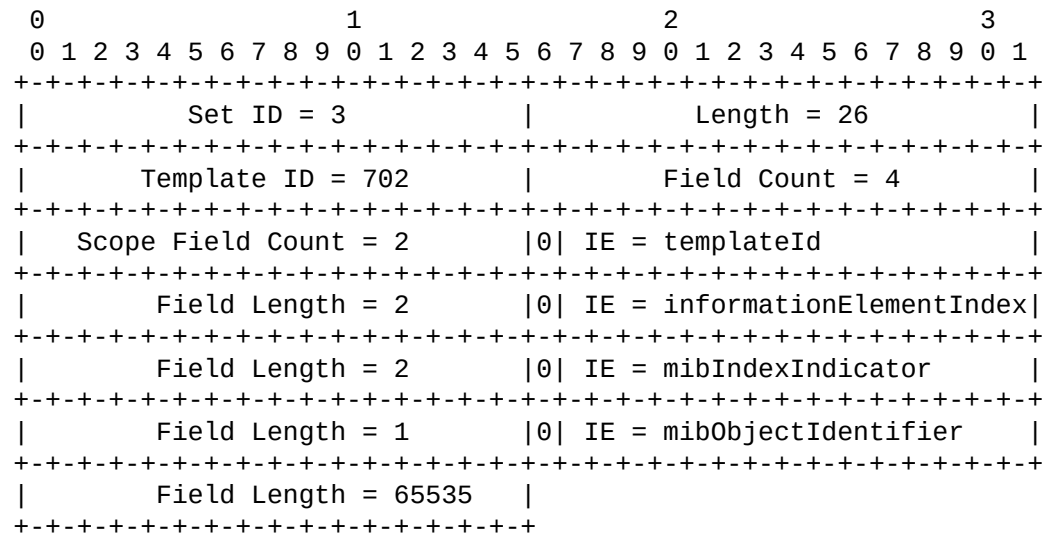


Figure 34: Example of a MIB Field Options Template for an Indexed MIB Object with Two Index Objects

Figure 35 shows the exported MIB Field Options Data used to export the required mibObjectValue Information Element metadata. Note that the first two Data Records have all their mibIndexIndicator bits set to 0. The third mibIndexIndicator has the value "00000011" to show that the first two fields in the Data Record are the INDEXes for this columnar object.

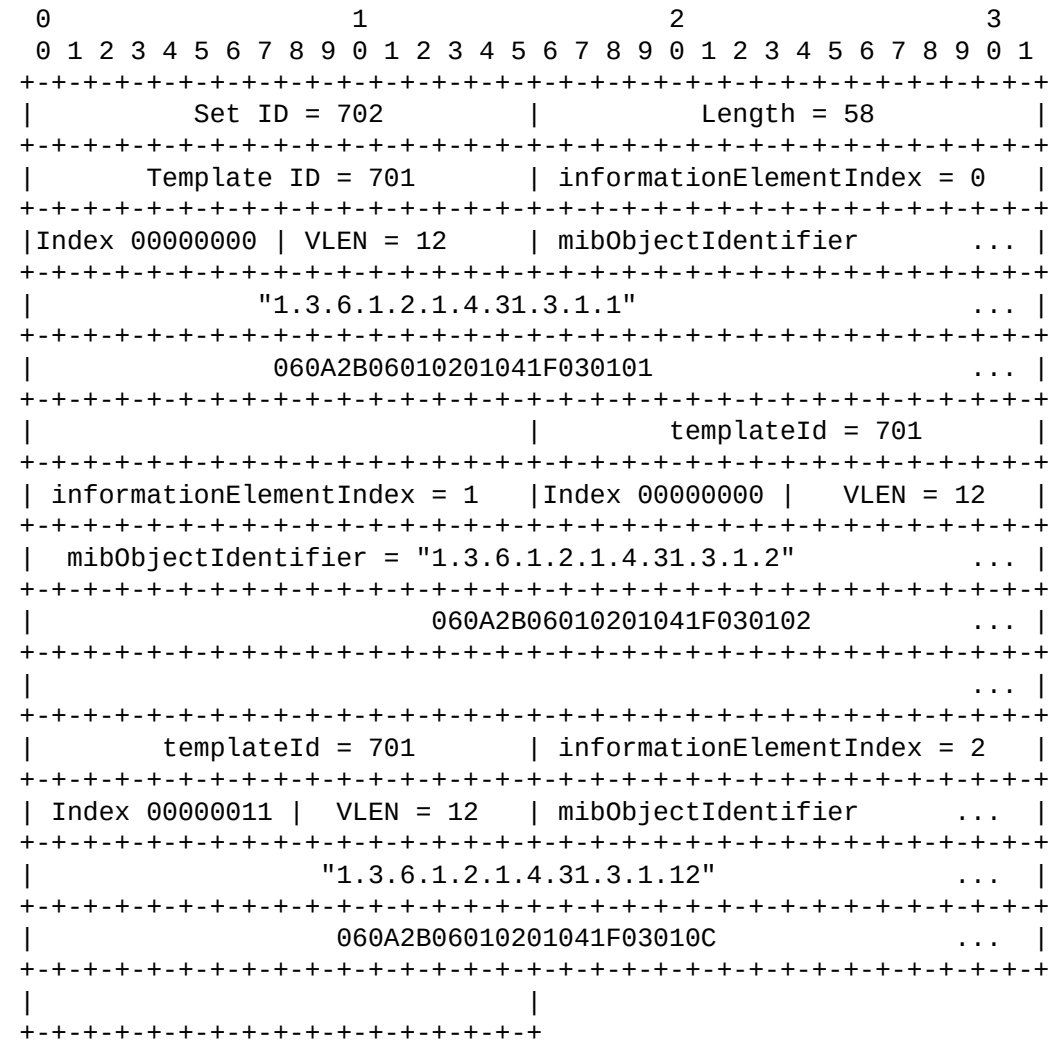


Figure 35: Example of a MIB Field Options Data Set for an Indexed MIB Object with Two Index Objects

Figure 36 shows the Data Records that export the values of the three mibObjectValue Information Elements.

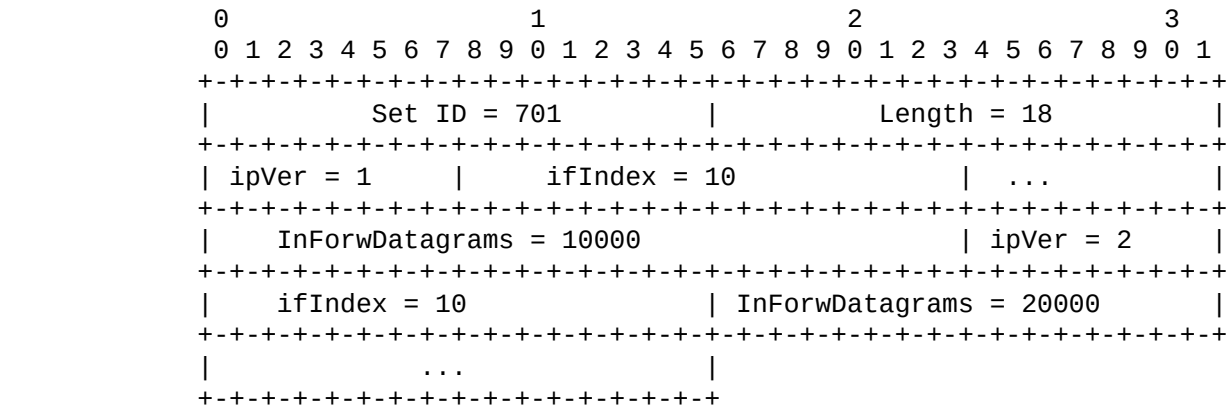


Figure 36: Example of a MIB Data Set for an Indexed MIB Object with Two Index Objects

6.6. Exporting a Columnar Object Indexed by Information Elements: ifOutQLen

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

The exported data looks like:

SRC ADDR	DST ADDR	PKT LEN	OUTPUT INTERFACE	OUTPUT QUEUE 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 8: Packet Report with Interface Output Queue Length (ifOutQLen) Data

The ifOutQLen MIB object, defined in the IF-MIB [RFC2863], provides the length of the output packet queue. This columnar object is part of the ifEntry conceptual row and indexed by the interface index (ifIndex).

This relationship between the ifOutQLen field and the index field is exported using mibIndexIndicator in the MIB Field Options Template. The value of "00001000" flags the index fields concisely.

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 37 shows the exported Template Set detailing the Template for exporting a PSAMP Report with ifOutQLen. Figures 38 and 39 show the MIB Field Options Template and Data Record.

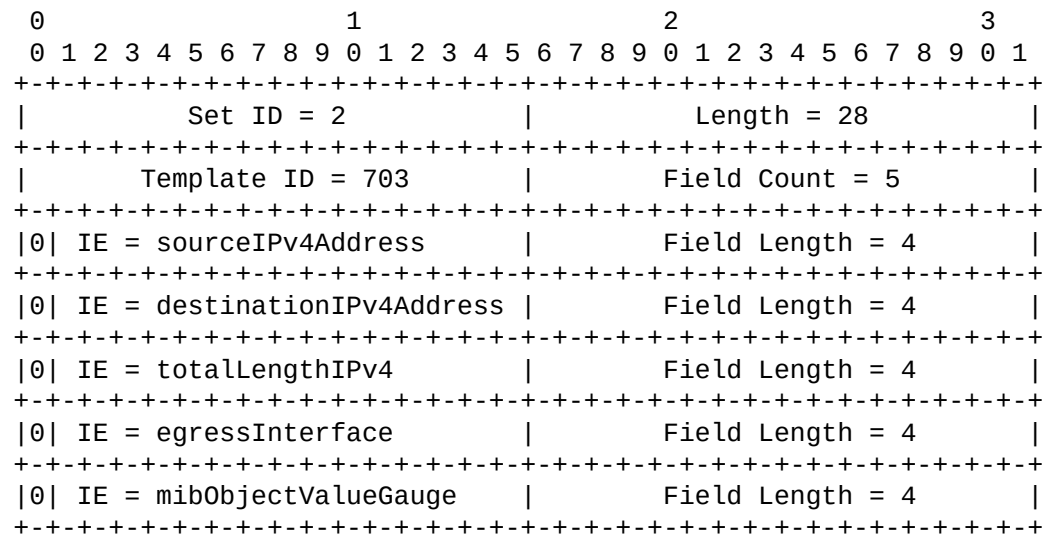


Figure 37: Example of Template for a PSAMP Report with ifOutQLen Indexed by egressInterface

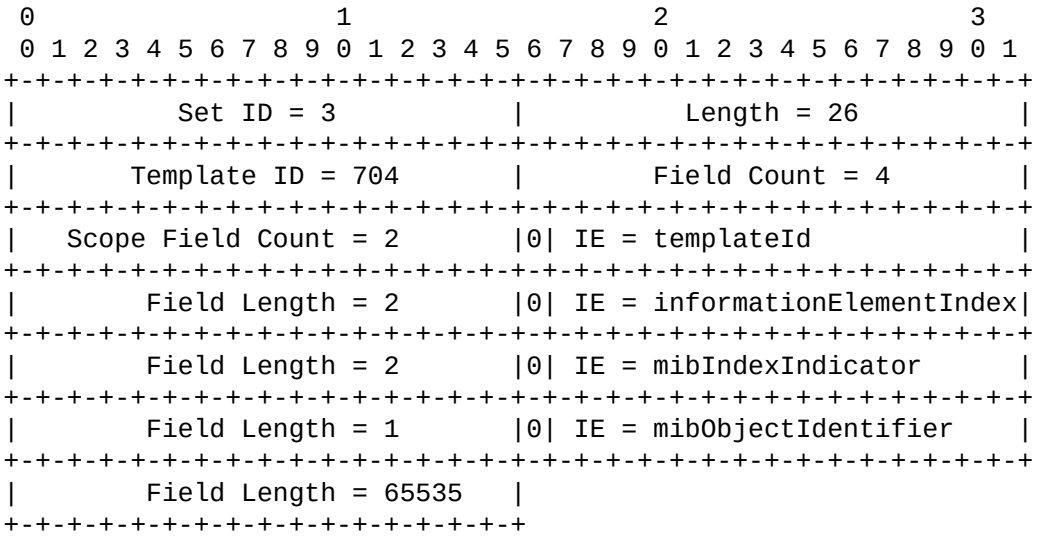


Figure 38: Example of MIB Field Options Template for a PSAMP Report with ifOutQLen Indexed by egressInterface

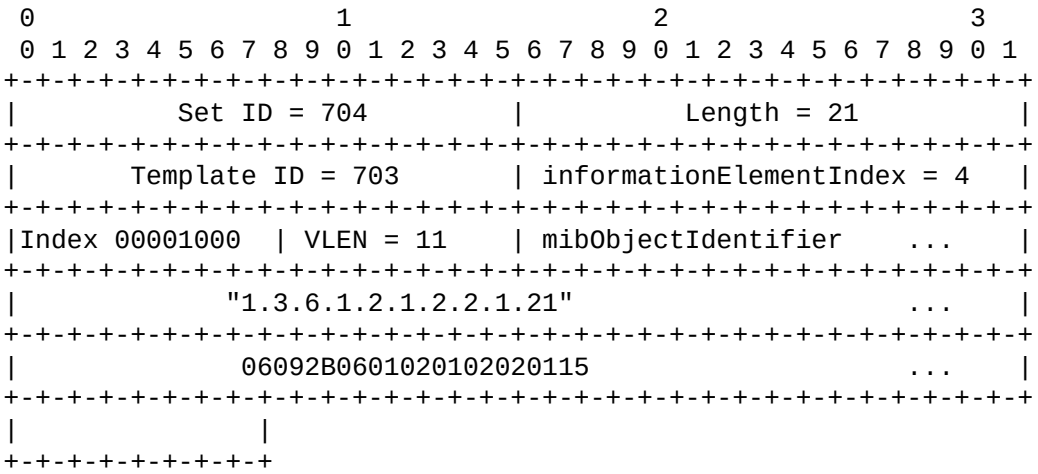


Figure 39: Example of MIB Field Options Data Record for a PSAMP Report with ifOutQLen Indexed by egressInterface

The corresponding IPFIX Data Record is shown in Figure 40. 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.

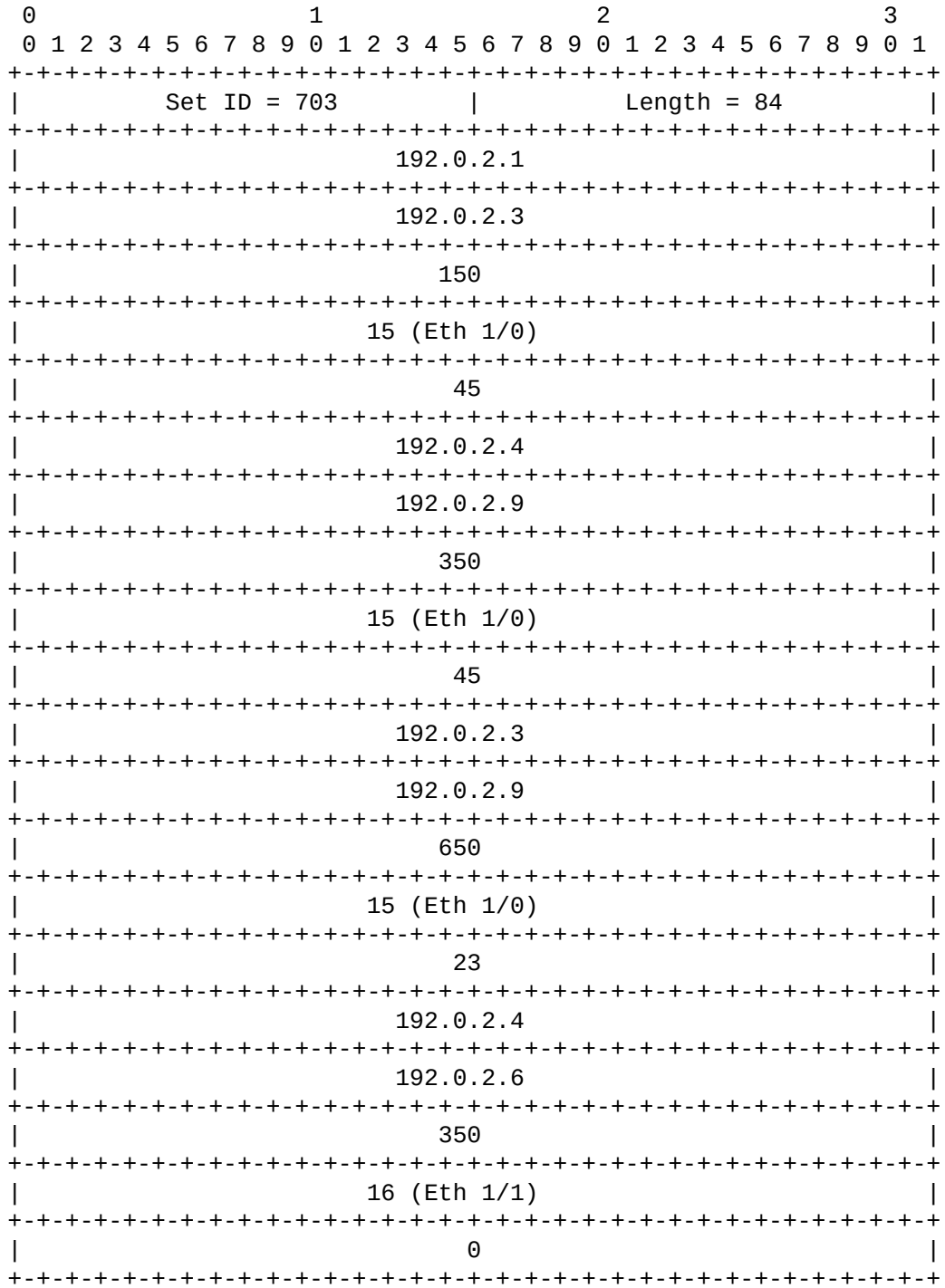


Figure 40: Example of PSAMP Packet Report with ifOutQLen Indexed by egressInterface

6.7. Exporting with Multiple Contexts: The OSPF Neighbor Row Revisited

If the context used to export the MIB objects is the default one, no extra context fields are required. This example demonstrates how to handle the case when the context needs to be specified. It is based on the previous example (Section 6.3).

The OSPF details of the conceptual row that was exported per Section 6.3 would be suitable if there were only one OSPF process running at the Observation Point. If multiple OSPF processes are present, then they can be differentiated by also exporting the mibContextEngineID and mibContextName.

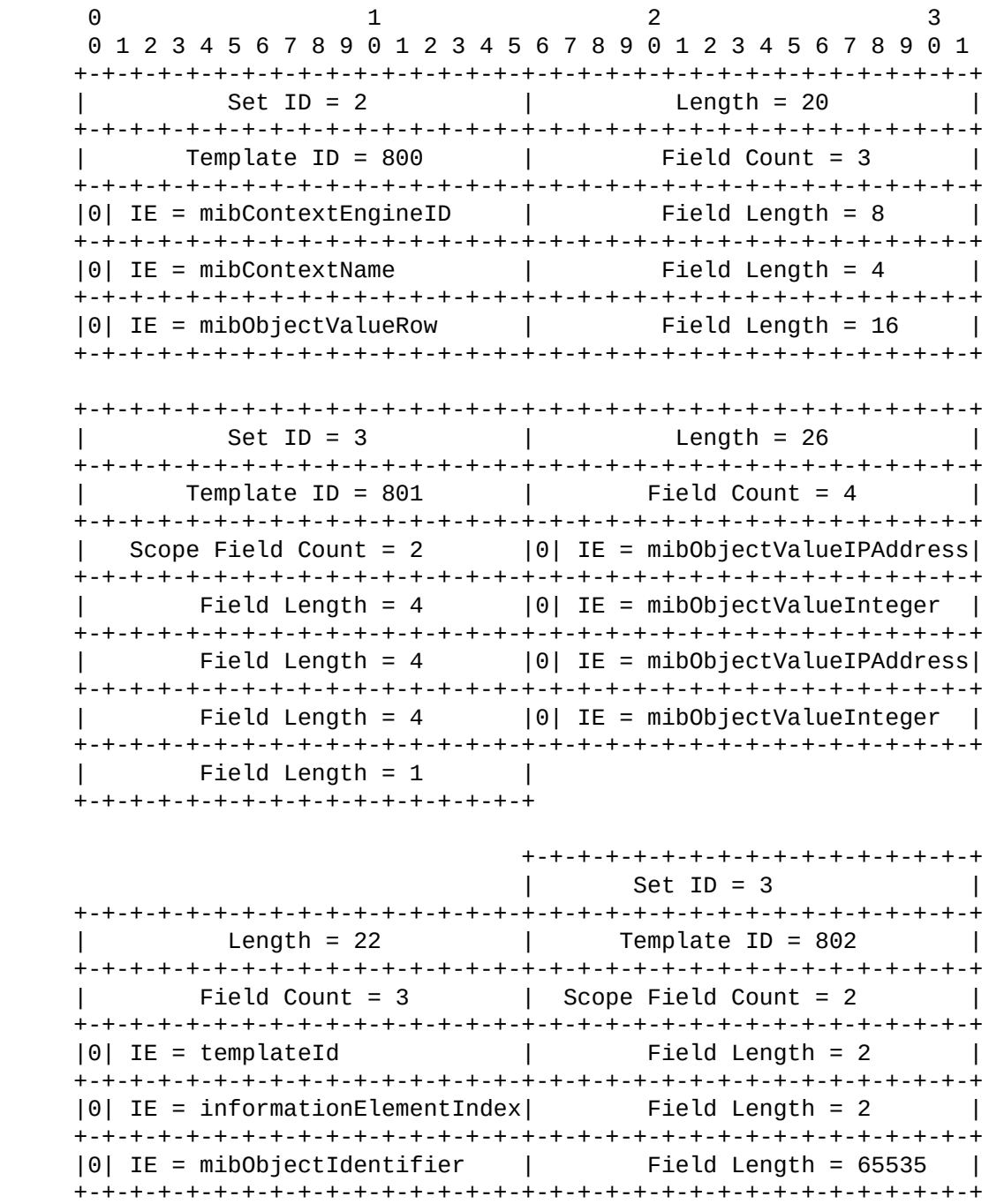
The following fields will be exported:

Object	ID	mibObjectValue	Len
ospfNbrIpAddr	ospfNbrEntry 1	mibObjectValueIPAddress	4
ospfNbrAddress-LessIndex	ospfNbrEntry 2	mibObjectValueInteger	4
ospfNbrRtrId	ospfNbrEntry 3	mibObjectValueIPAddress	4
ospfNbrState	ospfNbrEntry 6	mibObjectValueInteger	1

Table 9: OSPF Neighbor Entry Objects

The example contextEngineID matches the example from [RFC3411] for Acme Networks: "'800002B804616263'H (enterprise 696, string "abc")".

Figure 41 shows the Templates exported to support a mibObjectValueRow that is defined within a context. Figure 42 shows the example OID Data for the conceptual row exported in mibObjectValueRow. These are unchanged from the previous example (Section 6.3). Figure 43 shows the example data for two OSPF neighbors. Although these have identical INDEX/scope values, the context information indicates that they come from different OSPF processes. Note that the OID for ospfNbrEntry has been encoded using ASN.1/BER to "06082B060102010E0A01" at 10 octets long.


0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Set ID = 2
Length = 20
Template ID = 800
Field Count = 3
0| IE = mibContextEngineID
Field Length = 8
0| IE = mibContextName
Field Length = 4
0| IE = mibObjectValueRow
Field Length = 16
Set ID = 3
Length = 26
Template ID = 801
Field Count = 4
Scope Field Count = 2
0| IE = mibObjectValueIPAddress
Field Length = 4
0| IE = mibObjectValueInteger
Field Length = 4
0| IE = mibObjectValueIPAddress
Field Length = 4
0| IE = mibObjectValueInteger
Field Length = 1
Set ID = 3
Length = 22
Template ID = 802
Field Count = 3
Scope Field Count = 2
0| IE = templateId
Field Length = 2
0| IE = informationElementIndex
Field Length = 2
0| IE = mibObjectIdentifier
Field Length = 65535

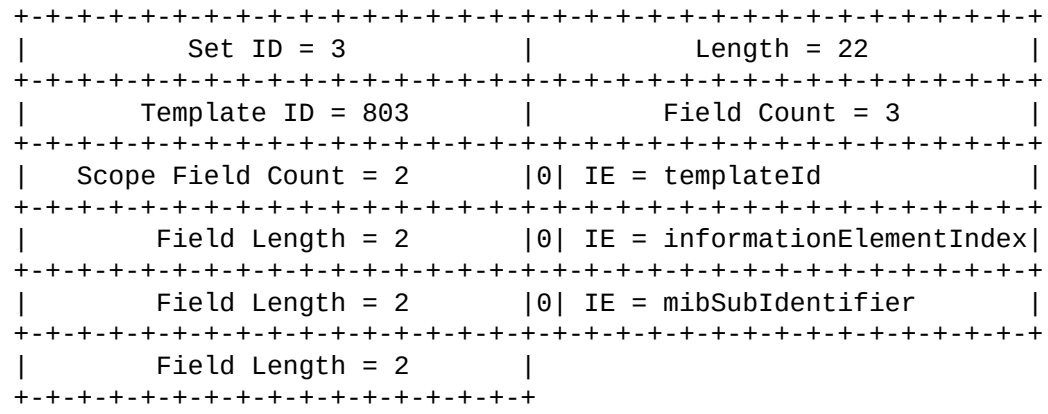


Figure 41: Example of ospfNbrEntry Template and Options Template Sets with Context

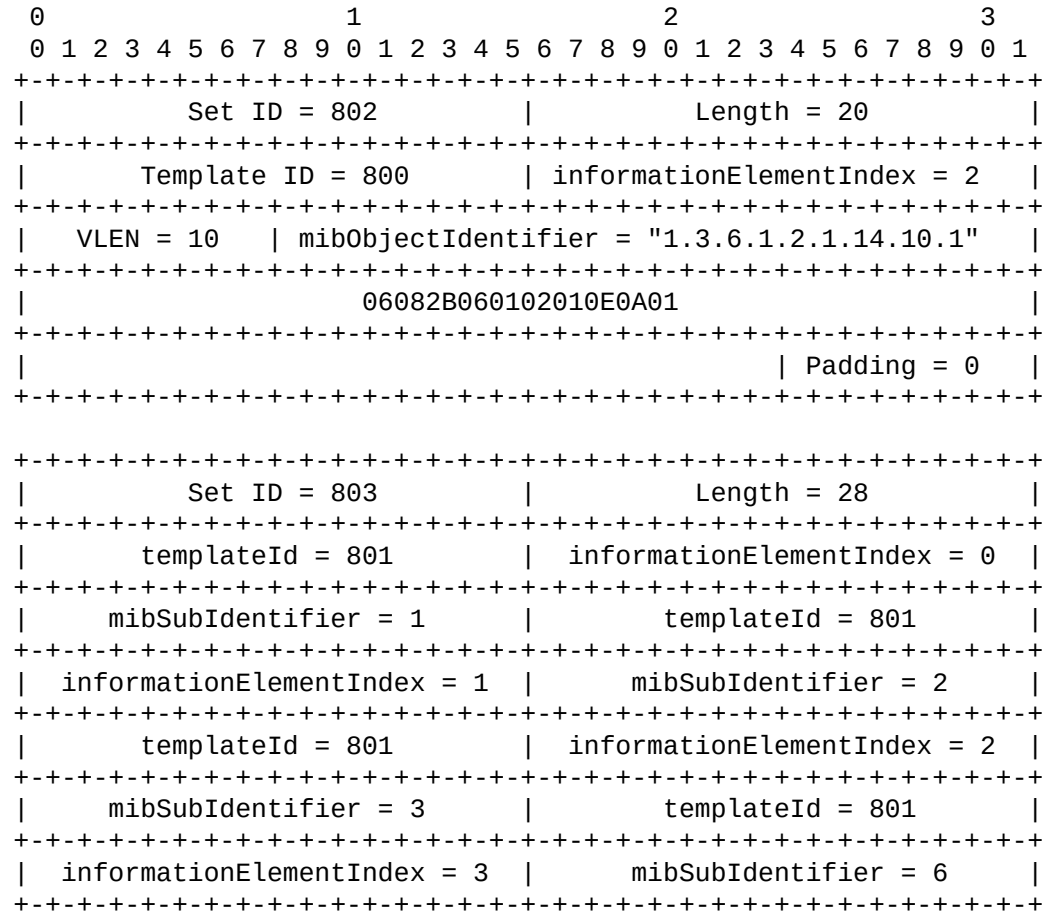


Figure 42: Example of ospfNbrEntry OID Data Export with Context

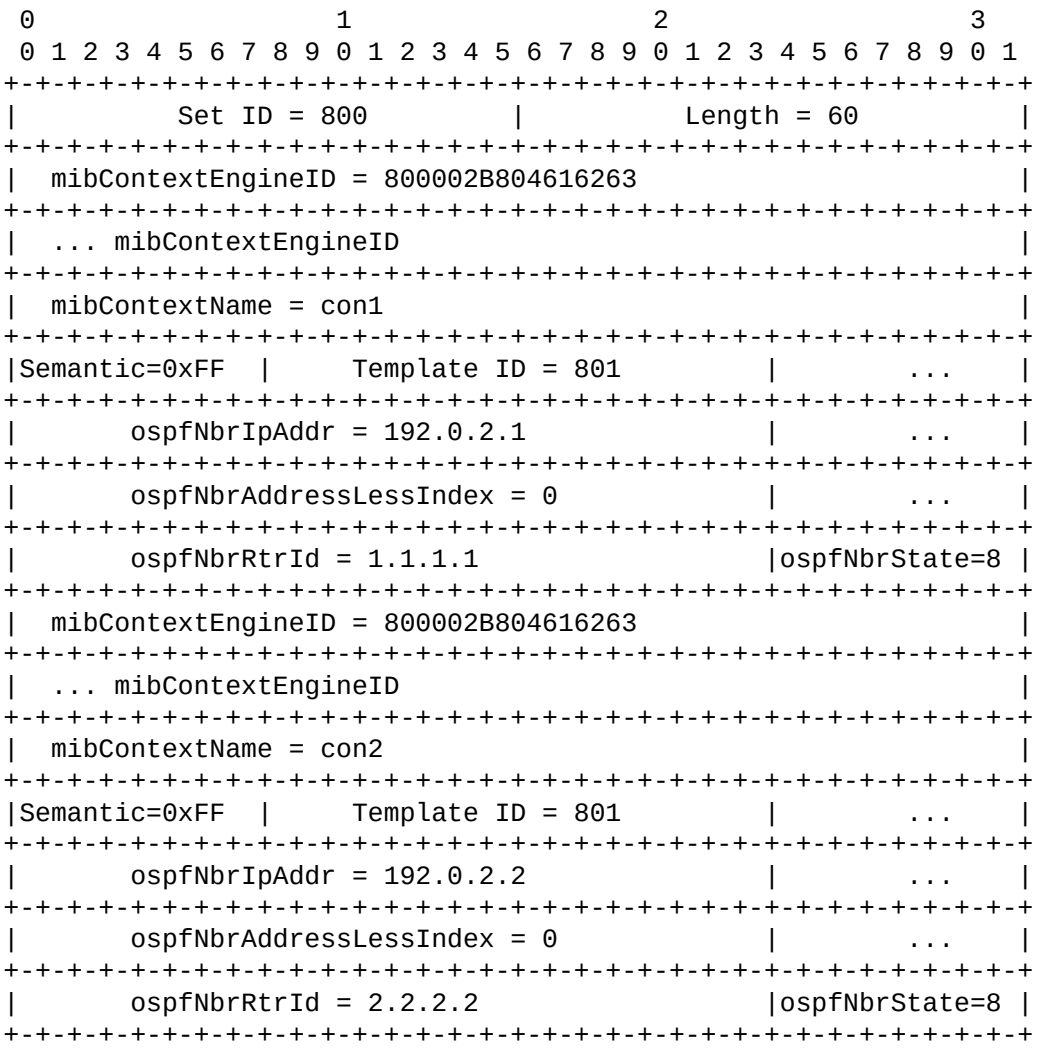


Figure 43: Example of Data Export for ospfNbrEntry with Context

7. Configuration Considerations

When configuring a MIB OID for export, consideration should be given to whether the SNMP context should also be configurable. If a non-default context is used, then it should be associated with the fields as per Section 5.6.

8. The Collecting Process's Side

The specifications in Section 9 of [RFC7011] also apply to Collectors that implement this specification. In addition, the following specifications should be noted:

- A Collecting Process that implements this specification MUST store the Data Records containing the OID object type definitions with the same retention policy as Templates.
- A Collecting Process that implements this specification SHOULD have access to MIB modules in order to look up the received MIB Object Identifiers and find the full type definition and name of MIB OID fields used in received Templates.
- It should be noted that, because reduced-size encoding MAY be used by the Exporting Process, the Collecting Process cannot assume that 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 MAY log a warning.
- A Collecting Process MUST support the three options for handling columnar objects detailed in Section 5.8.

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 monitoring applications periodically export a mapping of interface ID to interface name using IPFIX Options Templates. This could be expanded to include the ifInUcastPkts MIB object as defined in the IF-MIB [RFC2863], indexed using the ingressInterface Information Element. This would provide the input statistics for each interface; these statistics can be compared to the Flow information to ensure that the sampling rate is as expected, or, in the absence of sampling, to ensure that all expected packets are being monitored.

10. Security Considerations

For this extension to the IPFIX protocol, the same security considerations as those for the IPFIX protocol apply [RFC7011].

If the Exporter is generating or capturing the field values itself, e.g., using the MIB objects only as an encoding or type mechanism, there are no extra security considerations beyond standard IPFIX.

However, if the Exporter is implemented as an SNMP manager accessing an SNMP agent, it MUST authenticate itself to the SNMP agent [RFC3414] [RFC5591] [RFC5592] [RFC6353], and the SNMP agent MUST enforce SNMP access control rules [RFC3415] as required by the SNMP architecture [RFC3411].

Access to particular 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 that are included in IPFIX Data Records are objects that the receiving Collector is allowed to receive. Note that multiple users may have access to the data from the Collector.

When exporting MIB objects that may be considered sensitive or vulnerable in some network environments (as mentioned in the Security Considerations section of the RFC containing the MIB module), the Exporter should consider using anonymization techniques per [RFC6235] if the information is anonymizable. Consumers of exported data should therefore be able to handle the kinds of data modifications that are described in [RFC6235].

11. IANA Considerations

11.1. New IPFIX Semantics

New IPFIX semantics have been allocated in IANA's IPFIX registry [IANA-IPFIX] per Section 6 of [RFC7012], as defined in the subsections below.

11.1.1. snmpCounter

An integral value reporting the value of a counter, identical to the Counter32 and Counter64 semantics in [RFC2578], as determined by the Field Length.

This is similar to IPFIX's totalCounter semantic, except that total counters have an initial value of 0 but SNMP counters do not.

IANA has assigned value 7 to snmpCounter.

11.1.2. snmpGauge

An integral value identical to the Gauge32 semantic in [RFC2578] and the Gauge64 semantic in [RFC2856], as determined by the Field Length.

IANA has assigned value 8 to snmpGauge.

11.2. New IPFIX Information Elements

The new Information Elements in Table 10 have been allocated in IANA's IPFIX registry [IANA-IPFIX], as defined in the subsections below.

In each case, the "Units" and "Range" have been left blank, since these are not applicable.

ElementId	Name
434	mibObjectValueInteger
435	mibObjectValueOctetString
436	mibObjectValueOID
437	mibObjectValueBits
438	mibObjectValueIPAddress
439	mibObjectValueCounter
440	mibObjectValueGauge
441	mibObjectValueTimeTicks
442	mibObjectValueUnsigned
443	mibObjectValueTable
444	mibObjectValueRow
445	mibObjectIdentifier
446	mibSubIdentifier
447	mibIndexIndicator
448	mibCaptureTimeSemantics
449	mibContextEngineID
450	mibContextName
451	mibObjectName
452	mibObjectDescription
453	mibObjectSyntax
454	mibModuleName

Table 10: New Information Elements

11.2.1. New MIB Object Value Information Elements

11.2.1.1. mibObjectValueInteger

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

> Description: An IPFIX Information Element that denotes that the integer value of a MIB object will be exported. The MIB Object Identifier ("mibObjectIdentifier") for this field MUST be exported in a MIB Field Option or via another means. This Information

Element is used for MIB objects with the Base syntax of Integer32 and INTEGER with IPFIX reduced-size encoding used as required. The value is encoded as per the standard IPFIX Abstract Data Type of signed32.

Abstract Data Type: signed32

Data Type Semantics: quantity

ElementId: 434

Status: current

Reference: RFC 8038

11.2.1.2. mibObjectValueOctetString

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

Description: An IPFIX Information Element that denotes that an Octet String or Opaque value of a MIB object will be exported. The MIB Object Identifier ("mibObjectIdentifier") for this field MUST be exported in a MIB Field Option or via another means. This Information Element 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

Data Type Semantics: default

ElementId: 435

Status: current

Reference: RFC 8038

11.2.1.3. mibObjectValueOID

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

Description: An IPFIX Information Element that denotes that an Object Identifier or OID value of a MIB object will be exported. The MIB Object Identifier ("mibObjectIdentifier") for this field MUST be exported in a MIB Field Option or via another means. This

Information Element is used for MIB objects with the Base syntax of OBJECT IDENTIFIER. Note: In this case, the "mibObjectIdentifier" defines which MIB object is being exported, and the "mibObjectValueOID" field will contain the OID value of that MIB object. The mibObjectValueOID Information Element is encoded as ASN.1/BER [X.690] in an octetArray.

Abstract Data Type: octetArray

Data Type Semantics: default

ElementId: 436

Status: current

Reference: RFC 8038

11.2.1.4. mibObjectValueBits

A new Information Element "mibObjectValueBits" has been allocated in IANA's IPFIX registry [IANA-IPFIX], with the following definition:

Description: An IPFIX Information Element that denotes that a set of Enumerated flags or bits from a MIB object will be exported. The MIB Object Identifier ("mibObjectIdentifier") for this field MUST be exported in a MIB Field Option or via another means. This Information Element is used for MIB objects with the Base syntax of BITS. The flags or bits are encoded as per the standard IPFIX Abstract Data Type of octetArray, with sufficient length to accommodate the required number of bits. If the number of bits is not an integer multiple of octets, then the most significant bits at the end of the octetArray MUST be set to 0.

Abstract Data Type: octetArray

Data Type Semantics: flags

ElementId: 437

Status: current

Reference: RFC 8038

11.2.1.5. mibObjectValueIPAddress

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

> Description: An IPFIX Information Element that denotes that the IPv4 address value of a MIB object will be exported. The MIB Object Identifier ("mibObjectIdentifier") for this field MUST be exported in a MIB Field Option or via another means. This Information Element 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
>
> Data Type Semantics: default
>
> ElementId: 438
>
> Status: current
>
> Reference: RFC 8038

11.2.1.6. mibObjectValueCounter

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

> Description: An IPFIX Information Element that denotes that the counter value of a MIB object will be exported. The MIB Object Identifier ("mibObjectIdentifier") for this field MUST be exported in a MIB Field Option or via another means. This Information Element is used for MIB objects with the Base syntax of Counter32 or Counter64 with IPFIX reduced-size encoding used as required. The value is encoded as per the standard IPFIX Abstract Data Type of unsigned64.
>
> Abstract Data Type: unsigned64
>
> Data Type Semantics: snmpCounter
>
> ElementId: 439
>
> Status: current
>
> Reference: RFC 8038

11.2.1.7. mibObjectValueGauge

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

> Description: An IPFIX Information Element that denotes that the Gauge value of a MIB object will be exported. The MIB Object Identifier ("mibObjectIdentifier") for this field MUST be exported in a MIB Field Option or via another means. This Information Element is used for MIB objects with the Base syntax of Gauge32. The value is encoded as per the standard IPFIX Abstract Data Type of unsigned32. This value represents a non-negative integer that may increase or decrease but that shall never exceed a maximum value or fall below a minimum value.
>
> Abstract Data Type: unsigned32
>
> Data Type Semantics: snmpGauge
>
> ElementId: 440
>
> Status: current
>
> Reference: RFC 8038

11.2.1.8. mibObjectValueTimeTicks

A new Information Element "mibObjectValueTimeTicks" has been allocated in IANA's IPFIX registry [IANA-IPFIX], with the following definition:

> Description: An IPFIX Information Element that denotes that the TimeTicks value of a MIB object will be exported. The MIB Object Identifier ("mibObjectIdentifier") for this field MUST be exported in a MIB Field Option or via another means. This Information Element is used for MIB objects with the Base syntax of TimeTicks. The value is encoded as per the standard IPFIX Abstract Data Type of unsigned32.
>
> Abstract Data Type: unsigned32
>
> Data Type Semantics: quantity
>
> ElementId: 441
>
> Status: current
>
> Reference: RFC 8038

11.2.1.9. mibObjectValueUnsigned

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

> Description: An IPFIX Information Element that denotes that an unsigned integer value of a MIB object will be exported. The MIB Object Identifier ("mibObjectIdentifier") for this field MUST be exported in a MIB Field Option or via another means. This Information Element is used for MIB objects with the Base syntax of unsigned32 with IPFIX reduced-size encoding used as required. The value is encoded as per the standard IPFIX Abstract Data Type of unsigned32.
>
> Abstract Data Type: unsigned32
>
> Data Type Semantics: quantity
>
> ElementId: 442
>
> Status: current
>
> Reference: RFC 8038

11.2.1.10. mibObjectValueTable

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

> Description: An IPFIX Information Element that denotes that a complete or partial conceptual table will be exported. The MIB Object Identifier ("mibObjectIdentifier") for this field MUST be exported in a MIB Field Option or via another means. This Information Element is used for MIB objects with a syntax of SEQUENCE OF. This is encoded as a subTemplateList of mibObjectValue Information Elements. The Template specified in the subTemplateList MUST be an Options Template and MUST include all the objects listed in the INDEX clause as Scope Fields.
>
> Abstract Data Type: subTemplateList
>
> Data Type Semantics: list
>
> ElementId: 443
>
> Status: current
>
> Reference: RFC 8038

11.2.1.11. mibObjectValueRow

A new Information Element "mibObjectValueRow" has been allocated in IANA's IPFIX registry [IANA-IPFIX], with the following definition:

> Description: An IPFIX Information Element that denotes that a single row of a conceptual table will be exported. The MIB Object Identifier ("mibObjectIdentifier") for this field MUST be exported in a MIB Field Option or via another means. This Information Element is used for MIB objects with a syntax of SEQUENCE. This is encoded as a subTemplateList of mibObjectValue Information Elements. The subTemplateList exported MUST contain exactly one row (i.e., one instance of the subTemplate). The Template specified in the subTemplateList MUST be an Options Template and MUST include all the objects listed in the INDEX clause as Scope Fields.
>
> Abstract Data Type: subTemplateList
>
> Data Type Semantics: list
>
> ElementId: 444
>
> Status: current
>
> Reference: RFC 8038

11.2.2. New MIB Field Options Information Elements

11.2.2.1. mibObjectIdentifier

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

> Description: An IPFIX Information Element that denotes that a MIB Object Identifier (MIB OID) is exported in the (Options) Template Record. The mibObjectIdentifier Information Element contains the OID assigned to the MIB object type definition encoded as ASN.1/BER [X.690].
>
> Abstract Data Type: octetArray
>
> Data Type Semantics: default
>
> ElementId: 445

Status: current

Reference: RFC 8038

11.2.2.2. mibSubIdentifier

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

Description: A non-negative sub-identifier of an Object Identifier (OID).

Abstract Data Type: unsigned32

Data Type Semantics: identifier

ElementId: 446

Status: current

Reference: RFC 8038

11.2.2.3. mibIndexIndicator

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

Description: A set of bit fields that is used for marking the Information Elements of a Data Record that serve as INDEX MIB objects for an indexed columnar MIB object. Each bit represents an Information Element in the Data Record, with the n-th least significant bit representing the n-th Information Element. A bit set to 1 indicates that the corresponding Information Element is an index of the columnar object represented by the mibObjectValue. A bit set to 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 extra bits in the mibIndexIndicator for which no corresponding Information Element exists MUST have the value 0 and must be disregarded by the Collector. This Information Element may be exported with IPFIX reduced-size encoding.

Abstract Data Type: unsigned64

Data Type Semantics: flags

ElementId: 447

Status: current

Reference: RFC 8038

11.2.2.4. mibCaptureTimeSemantics

A new Information Element "mibCaptureTimeSemantics" has been 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 retrieved from the MIB 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 refers to the Timestamp fields included in the same Data Record. This field SHOULD be used when exporting a mibObjectValue that specifies counters or statistics.

If the MIB value was sampled by SNMP prior to the IPFIX Metering Process or Exporting Process retrieving the value (i.e., the data is already stale) and it is important to know the exact sampling time, then an additional observationTime* element should be paired with the OID using IPFIX Structured Data [RFC6313]. Similarly, if different MIB capture times apply to different mibObjectValue elements within the Data Record, then individual mibCaptureTimeSemantics Information Elements should be paired with each OID using IPFIX Structured Data.

Values:

0 undefined

1 begin - The value for the MIB object is captured from the MIB when the Flow is first observed

2 end - The value for the MIB object is captured from the MIB when the Flow ends

3 export - The value for the MIB object is captured from the MIB at export time

4 average - The value for the MIB object is an average of multiple captures from the MIB over the observed life of the Flow

Abstract Data Type: unsigned8

Data Type Semantics: identifier

ElementId: 448

Status: current

Reference: RFC 8038

11.2.2.5. mibContextEngineID

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

Description: A mibContextEngineID that specifies the SNMP engine ID for a MIB field being exported over IPFIX. Definition as per [RFC3411], Section 3.3.

Abstract Data Type: octetArray

Data Type Semantics: default

ElementId: 449

Status: current

Reference: RFC 8038

11.2.2.6. mibContextName

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

Description: An Information Element that 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

Data Type Semantics: default

ElementId: 450

Status: current

Reference: RFC 8038

11.2.3. New MIB Type Information Elements

11.2.3.1. mibObjectName

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

Description: The name (called a descriptor in [RFC2578]) of an object type definition.

Abstract Data Type: string

Data Type Semantics: default

ElementId: 451

Status: current

Reference: RFC 8038

11.2.3.2. mibObjectDescription

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

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

Abstract Data Type: string

Data Type Semantics: default

ElementId: 452

Status: current

Reference: RFC 8038

11.2.3.3. mibObjectSyntax

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

Description: The value of the SYNTAX clause of a MIB object type definition, which may include a textual convention or sub-typing. See [RFC2578].

Abstract Data Type: string

Data Type Semantics: default

ElementId: 453

Status: current

Reference: RFC 8038

11.2.3.4. mibModuleName

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

Description: The textual name of the MIB module that defines a MIB object.

Abstract Data Type: string

Data Type Semantics: default

ElementId: 454

Status: current

Reference: RFC 8038

12. References

12.1. Normative References

[IANA-IPFIX] IANA, "IP Flow Information Export (IPFIX) Entities", <http://www.iana.org/assignments/ipfix/>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2578] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, DOI 10.17487/RFC2578, April 1999, <http://www.rfc-editor.org/info/rfc2578>.

[RFC2856] Bierman, A., McCloghrie, K., and R. Presuhn, "Textual Conventions for Additional High Capacity Data Types", RFC 2856, DOI 10.17487/RFC2856, June 2000, <http://www.rfc-editor.org/info/rfc2856>.

[RFC3411] Harrington, D., Presuhn, R., and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", STD 62, RFC 3411, DOI 10.17487/RFC3411, December 2002, <http://www.rfc-editor.org/info/rfc3411>.

[RFC6526] Claise, B., Aitken, P., Johnson, A., and G. Muenz, "IP Flow Information Export (IPFIX) Per Stream Control Transmission Protocol (SCTP) Stream", RFC 6526, DOI 10.17487/RFC6526, March 2012, <http://www.rfc-editor.org/info/rfc6526>.

[RFC7011] Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, DOI 10.17487/RFC7011, September 2013, <http://www.rfc-editor.org/info/rfc7011>.

[RFC7012] Claise, B., Ed., and B. Trammell, Ed., "Information Model for IP Flow Information Export (IPFIX)", RFC 7012, DOI 10.17487/RFC7012, September 2013, <http://www.rfc-editor.org/info/rfc7012>.

[X.690] International Telecommunication Union, "Information Technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ITU-T Recommendation X.690, ISO/IEC 8825-1, August 2015, <https://www.itu.int/rec/T-REC-X.690>.

12.2. Informative References

[CISCO-PROCESS-MIB] Cisco Systems Inc., "CISCO-PROCESS-MIB.my: MIB for CPU and process statistics", <ftp://ftp.cisco.com/pub/mibs/v2/CISCO-PROCESS-MIB.my>.

[RFC2863] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB", RFC 2863, DOI 10.17487/RFC2863, June 2000, <http://www.rfc-editor.org/info/rfc2863>.

[RFC2982] Kavasseri, R., Ed., "Distributed Management Expression MIB", RFC 2982, DOI 10.17487/RFC2982, October 2000, <http://www.rfc-editor.org/info/rfc2982>.

[RFC3414] Blumenthal, U. and B. Wijnen, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)", STD 62, RFC 3414, DOI 10.17487/RFC3414, December 2002, <http://www.rfc-editor.org/info/rfc3414>.

[RFC3415] Wijnen, B., Presuhn, R., and K. McCloghrie, "View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3415, DOI 10.17487/RFC3415, December 2002, <http://www.rfc-editor.org/info/rfc3415>.

[RFC3444] Pras, A. and J. Schoenwaelder, "On the Difference between Information Models and Data Models", RFC 3444, DOI 10.17487/RFC3444, January 2003, <http://www.rfc-editor.org/info/rfc3444>.

[RFC4022] Raghunarayan, R., Ed., "Management Information Base for the Transmission Control Protocol (TCP)", RFC 4022, DOI 10.17487/RFC4022, March 2005, <http://www.rfc-editor.org/info/rfc4022>.

[RFC4293] Routhier, S., Ed., "Management Information Base for the Internet Protocol (IP)", RFC 4293, DOI 10.17487/RFC4293, April 2006, <http://www.rfc-editor.org/info/rfc4293>.

[RFC4750] Joyal, D., Ed., Galecki, P., Ed., Giacalone, S., Ed., Coltun, R., and F. Baker, "OSPF Version 2 Management Information Base", RFC 4750, DOI 10.17487/RFC4750, December 2006, <http://www.rfc-editor.org/info/rfc4750>.

[RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <http://www.rfc-editor.org/info/rfc4960>.

[RFC5102] Quittek, J., Bryant, S., Claise, B., Aitken, P., and J. Meyer, "Information Model for IP Flow Information Export", RFC 5102, DOI 10.17487/RFC5102, January 2008, <http://www.rfc-editor.org/info/rfc5102>.

[RFC5476] Claise, B., Ed., Johnson, A., and J. Quittek, "Packet Sampling (PSAMP) Protocol Specifications", RFC 5476, DOI 10.17487/RFC5476, March 2009, <http://www.rfc-editor.org/info/rfc5476>.

[RFC5591] Harrington, D. and W. Hardaker, "Transport Security Model for the Simple Network Management Protocol (SNMP)", STD 78, RFC 5591, DOI 10.17487/RFC5591, June 2009, <http://www.rfc-editor.org/info/rfc5591>.

[RFC5592] Harrington, D., Salowey, J., and W. Hardaker, "Secure Shell Transport Model for the Simple Network Management Protocol (SNMP)", RFC 5592, DOI 10.17487/RFC5592, June 2009, <http://www.rfc-editor.org/info/rfc5592>.

[RFC6235] Boschi, E. and B. Trammell, "IP Flow Anonymization Support", RFC 6235, DOI 10.17487/RFC6235, May 2011, <http://www.rfc-editor.org/info/rfc6235>.

[RFC6313] Claise, B., Dhandapani, G., Aitken, P., and S. Yates, "Export of Structured Data in IP Flow Information Export (IPFIX)", RFC 6313, DOI 10.17487/RFC6313, July 2011, <http://www.rfc-editor.org/info/rfc6313>.

[RFC6353] Hardaker, W., "Transport Layer Security (TLS) Transport Model for the Simple Network Management Protocol (SNMP)", STD 78, RFC 6353, DOI 10.17487/RFC6353, July 2011, <http://www.rfc-editor.org/info/rfc6353>.

Acknowledgments

The authors would like to thank Andrew Johnson for his collaboration on the first draft version of this document, and to thank Andrew Feren and Brian Trammell for their detailed reviews.

Juergen Schoenwaelder was partly funded by Flamingo, a Network of Excellence project (ICT-318488) supported by the European Commission under its Seventh Framework Programme.

Authors' Addresses

Paul Aitken (editor)
Brocade Communications Systems, Inc.
19a Canning Street, Level 3
Edinburgh, Scotland EH3 8EG
United Kingdom

Phone: +44 203 005 0731
Email: paitken@brocade.com

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

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

Srikar B S
Mojo Networks, Inc.
S. No. 7, Pinnac House II
Kothrud, Pune 411038
India

Phone: +91 94 4847 6672
Email: srikarbs@gmail.com

Colin McDowall
Brocade Communications Systems, Inc.
19a Canning Street, Level 3
Edinburgh, Scotland EH3 8EG
United Kingdom

Phone: +44 203 005 0687
Email: cmcdowal@brocade.com

Juergen Schoenwaelder
Jacobs University Bremen
Campus Ring 1
Bremen 28725
Germany

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