

CybOX™ Version 2.1.1. Part 37: Network Flow Object

Committee Specification Draft 01 / Public Review Draft 01

20 June 2016

Specification URIs

This version:

<http://docs.oasis-open.org/cti/cybox/v2.1.1/csprd01/part37-network-flow/cybox-v2.1.1-csprd01-part37-network-flow.docx> (Authoritative)
<http://docs.oasis-open.org/cti/cybox/v2.1.1/csprd01/part37-network-flow/cybox-v2.1.1-csprd01-part37-network-flow.html>
<http://docs.oasis-open.org/cti/cybox/v2.1.1/csprd01/part37-network-flow/cybox-v2.1.1-csprd01-part37-network-flow.pdf>

Previous version:

N/A

Latest version:

<http://docs.oasis-open.org/cti/cybox/v2.1.1/part37-network-flow/cybox-v2.1.1-part37-network-flow.docx> (Authoritative)
<http://docs.oasis-open.org/cti/cybox/v2.1.1/part37-network-flow/cybox-v2.1.1-part37-network-flow.html>
<http://docs.oasis-open.org/cti/cybox/v2.1.1/part37-network-flow/cybox-v2.1.1-part37-network-flow.pdf>

Technical Committee:

OASIS Cyber Threat Intelligence (CTI) TC

Chair:

Richard Struse (Richard.Struse@HQ.DHS.GOV), DHS Office of Cybersecurity and Communications (CS&C)

Editors:

Desiree Beck (dbeck@mitre.org), MITRE Corporation
Trey Darley (trey@kingfisherops.com), Individual member
Ivan Kirillov (ikirillov@mitre.org), MITRE Corporation
Rich Piazza (rpiazza@mitre.org), MITRE Corporation

Additional artifacts:

This prose specification is one component of a Work Product whose components are listed in <http://docs.oasis-open.org/cti/cybox/v2.1.1/csprd01/cybox-v2.1.1-csprd01-additional-artifacts.html>.

Related work:

This specification is related to:

- *STIX™ Version 1.2.1*. Edited by Sean Barnum, Desiree Beck, Aharon Chernin, and Rich Piazza. 05 May 2016. OASIS Committee Specification 01. <http://docs.oasis-open.org/cti/stix/v1.2.1/cs01/part1-overview/stix-v1.2.1-cs01-part1-overview.html>.

Abstract:

The Cyber Observable Expression (CybOX) is a standardized language for encoding and communicating high-fidelity information about cyber observables, whether dynamic events or stateful measures that are observable in the operational cyber domain. By specifying a common structured schematic mechanism for these cyber observables, the intent is to enable the potential for detailed automatable sharing, mapping, detection and analysis heuristics. This specification document defines the Network Flow Object data model, which is one of the Object data models for CybOX content.

Status:

This document was last revised or approved by the OASIS Cyber Threat Intelligence (CTI) TC on the above date. The level of approval is also listed above. Check the "Latest version" location noted above for possible later revisions of this document. Any other numbered Versions and other technical work produced by the Technical Committee (TC) are listed at https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=cti#technical.

TC members should send comments on this specification to the TC's email list. Others should send comments to the TC's public comment list, after subscribing to it by following the instructions at the "[Send A Comment](#)" button on the TC's web page at <https://www.oasis-open.org/committees/cti/>.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the TC's web page (<https://www.oasis-open.org/committees/cti/ipr.php>).

Citation format:

When referencing this specification the following citation format should be used:

[CybOX-v2.1.1-network-flow]

CybOX™ Version 2.1.1. Part 37: Network Flow Object. Edited by Desiree Beck, Trey Darley, Ivan Kirillov, and Rich Piazza. 20 June 2016. OASIS Committee Specification Draft 01 / Public Review Draft 01. <http://docs.oasis-open.org/cti/cybox/v2.1.1/csprd01/part37-network-flow/cybox-v2.1.1-csprd01-part37-network-flow.html>. Latest version: <http://docs.oasis-open.org/cti/cybox/v2.1.1/part37-network-flow/cybox-v2.1.1-part37-network-flow.html>.

Notices

Copyright © OASIS Open 2016. All Rights Reserved.

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full [Policy](#) may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The name "OASIS" is a trademark of [OASIS](#), the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see <https://www.oasis-open.org/policies-guidelines/trademark> for above guidance.

Portions copyright © United States Government 2012-2016. All Rights Reserved.

STIX™, TAXII™, AND CyBOX™ (STANDARD OR STANDARDS) AND THEIR COMPONENT PARTS ARE PROVIDED "AS IS" WITHOUT ANY WARRANTY OF ANY KIND, EITHER EXPRESSED, IMPLIED, OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, ANY WARRANTY THAT THESE STANDARDS OR ANY OF THEIR COMPONENT PARTS WILL CONFORM TO SPECIFICATIONS, ANY IMPLIED

WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR FREEDOM FROM INFRINGEMENT, ANY WARRANTY THAT THE STANDARDS OR THEIR COMPONENT PARTS WILL BE ERROR FREE, OR ANY WARRANTY THAT THE DOCUMENTATION, IF PROVIDED, WILL CONFORM TO THE STANDARDS OR THEIR COMPONENT PARTS. IN NO EVENT SHALL THE UNITED STATES GOVERNMENT OR ITS CONTRACTORS OR SUBCONTRACTORS BE LIABLE FOR ANY DAMAGES, INCLUDING, BUT NOT LIMITED TO, DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES, ARISING OUT OF, RESULTING FROM, OR IN ANY WAY CONNECTED WITH THESE STANDARDS OR THEIR COMPONENT PARTS OR ANY PROVIDED DOCUMENTATION, WHETHER OR NOT BASED UPON WARRANTY, CONTRACT, TORT, OR OTHERWISE, WHETHER OR NOT INJURY WAS SUSTAINED BY PERSONS OR PROPERTY OR OTHERWISE, AND WHETHER OR NOT LOSS WAS SUSTAINED FROM, OR AROSE OUT OF THE RESULTS OF, OR USE OF, THE STANDARDS, THEIR COMPONENT PARTS, AND ANY PROVIDED DOCUMENTATION. THE UNITED STATES GOVERNMENT DISCLAIMS ALL WARRANTIES AND LIABILITIES REGARDING THE STANDARDS OR THEIR COMPONENT PARTS ATTRIBUTABLE TO ANY THIRD PARTY, IF PRESENT IN THE STANDARDS OR THEIR COMPONENT PARTS AND DISTRIBUTES IT OR THEM "AS IS."

Table of Contents

1	Introduction	7
1.1	CybOX™ Specification Documents	7
1.2	Document Conventions	7
1.2.1	Fonts	7
1.2.2	UML Package References	8
1.2.3	UML Diagrams	8
1.2.4	Property Table Notation	9
1.2.5	Property and Class Descriptions	9
1.3	Terminology	10
1.4	Normative References	10
2	Background Information	11
2.1	Cyber Observables	11
2.2	Objects	11
3	Data Model	12
3.1	NetworkFlowObjectType Class	12
3.2	NetworkLayerInfoType Class	14
3.3	NetworkFlowLabelType Class	15
3.4	UnidirectionalRecordType Class	15
3.5	BidirectionalRecordType Class	17
3.6	IPFIXMessageType Class	18
3.6.1	IPFIXMessageHeaderType Class	19
3.6.2	IPFIXSetType Class	20
3.6.3	IPFIXTemplateSetType Class	23
3.6.4	IPFIXOptionsTemplateSetType Class	23
3.6.5	IPFIXDataSetType Class	24
3.6.6	IPFIXSetHeaderType Class	25
3.6.7	IPFIXTemplateRecordType Class	26
3.6.8	IPFIXTemplateRecordHeaderType Class	26
3.6.9	IPFIXTemplateRecordFieldSpecifiersType Class	27
3.6.10	IPFIXOptionsTemplateRecordType Class	28
3.6.11	IPFIXOptionsTemplateRecordHeaderType Class	29
3.6.12	IPFIXOptionsTemplateRecordFieldSpecifiersType Class	30
3.6.13	IPFIXDataRecordType Class	32
3.7	NetflowV9ExportPacketType Class	33
3.7.1	NetflowV9PacketHeaderType Class	34
3.7.2	NetflowV9FlowSetType Class	35
3.7.3	NetflowV9TemplateFlowSetType Class	37
3.7.4	NetflowV9TemplateRecordType Class	38
3.7.5	NetflowV9FieldType Data Type	38
3.7.6	NetflowV9OptionsTemplateFlowSetType Class	39
3.7.7	NetflowV9OptionsTemplateRecordType Class	39
3.7.8	NetflowV9ScopeFieldType Data Type	41
3.7.9	NetflowV9DataFlowSetType Class	41

3.7.10	NetflowV9DataRecordType Class	42
3.7.11	FlowDataRecordType Class	44
3.7.12	FlowCollectionElementType Class	44
3.7.13	OptionsDataRecordType Class	44
3.7.14	OptionCollectionElementType Class	45
3.8	NetflowV5PacketType Class	46
3.8.1	NetflowV5FlowHeaderType Class.....	47
3.8.2	NetflowV5FlowRecordType Class.....	48
3.9	SiLKRecordType Class.....	50
3.9.1	SiLKFlowAttributesType Data Type	53
3.9.2	SiLKAddressType Data Type	53
3.9.3	SiLKCountryCodeType Class.....	54
3.9.4	SiLKSensorInfoType Class	54
3.9.5	SiLKDirectionType Class.....	54
3.9.6	SiLKSensorClassType Class	55
3.10	YAFRecordType Class	55
3.10.1	YAFFlowType Class.....	56
3.10.2	YAFReverseFlowType Class	58
3.10.3	YAFTCPFlowType Class	60
3.11	NetflowV9FieldTypeEnum Enumeration.....	60
3.12	NetflowV9ScopeFieldTypeEnum Enumeration	63
3.13	SiLKFlowAttributesTypeEnum Enumeration	63
3.14	SiLKAddressTypeEnum Enumeration	64
3.15	SiLKDirectionTypeEnum Enumeration	64
3.16	SiLKSensorClassTypeEnum Enumeration	65
4	Conformance	66
Appendix A. Acknowledgments		67
Appendix B. Revision History.....		71

1 Introduction

[All text is normative unless otherwise labeled]

The Cyber Observable Expression (CybOX™) provides a common structure for representing cyber observables across and among the operational areas of enterprise cyber security. CybOX improves the consistency, efficiency, and interoperability of deployed tools and processes, and it increases overall situational awareness by enabling the potential for detailed automatable sharing, mapping, detection, and analysis heuristics.

This document serves as the specification for the CybOX Network Flow Object Version 2.1.1 data model, which is one of eighty-eight CybOX Object data models.

In Section 1.1, we discuss additional specification documents, in Section 1.2, we provide document conventions, and in Section 1.3, we provide terminology. References are given in Section 1.4. In Section 2, we give background information necessary to fully understand the Network Flow Object data model. We present the Network Flow Object data model specification details in Section 3 and conformance information in Section 4.

1.1 CybOX™ Specification Documents

The CybOX specification consists of a formal UML model and a set of textual specification documents that explain the UML model. Specification documents have been written for each of the individual data models that compose the full CybOX UML model.

CybOX has a modular design comprising two fundamental data models and a collection of Object data models. The fundamental data models – CybOX Core and CybOX Common – provide essential CybOX structure and functionality. The CybOX Objects, defined in individual data models, are precise characterizations of particular types of observable cyber entities (e.g., HTTP session, Windows registry key, DNS query).

Use of the CybOX Core and Common data models is required; however, use of the CybOX Object data models is purely optional: users select and use only those Objects and corresponding data models that are needed. Importing the entire CybOX suite of data models is not necessary.

The [CybOX Version 2.1.1 Part 1: Overview](#) document provides a comprehensive overview of the full set of CybOX data models, which in addition to the Core, Common, and numerous Object data models, includes various extension data models and a vocabularies data model, which contains a set of default controlled vocabularies. [CybOX Version 2.1.1 Part 1: Overview](#) also summarizes the relationship of CybOX to other languages, and outlines general CybOX data model conventions.

1.2 Document Conventions

The following conventions are used in this document.

1.2.1 Fonts

The following font and font style conventions are used in the document:

- Capitalization is used for CybOX high level concepts, which are defined in [CybOX Version 2.1.1 Part 1: Overview](#).

Examples: Action, Object, Event, Property

- The Courier New font is used for writing UML objects.

Examples: ActionType, cyboxCommon:BaseObjectPropertyType

Note that all high level concepts have a corresponding UML object. For example, the Action high level concept is associated with a UML class named, ActionType.

- The '*italic*' font (with single quotes) is used for noting actual, explicit values for CybOX Language properties. The *italic* font (without quotes) is used for noting example values.

Example: 'HashNameVocab-1.0,' *high, medium, low*

1.2.2 UML Package References

Each CybOX data model is captured in a different UML package (e.g., Core package) where the packages together compose the full CybOX UML model. To refer to a particular class of a specific package, we use the format `package_prefix:class`, where `package_prefix` corresponds to the appropriate UML package. The [CybOX Version 2.1.1 Part 1: Overview](#) document contains the full list of CybOX packages, along with the associated prefix notations, descriptions, and examples.

The `package_prefix` for the Network Flow data model is `NetFlowObj`. Note that in this specification document, we do not explicitly specify the package prefix for any classes that originate from the Network Flow Object data model.

1.2.3 UML Diagrams

This specification makes use of UML diagrams to visually depict relationships between CybOX Language constructs. Note that the diagrams have been extracted directly from the full UML model for CybOX; they have not been constructed purely for inclusion in the specification documents. Typically, diagrams are included for the primary class of a data model, and for any other class where the visualization of its relationships between other classes would be useful. This implies that there will be very few diagrams for classes whose only properties are either a data type or a class from the CybOX Common data model. Other diagrams that are included correspond to classes that specialize a superclass and abstract or generalized classes that are extended by one or more subclasses.

In UML diagrams, classes are often presented with their attributes elided, to avoid clutter. The fully described class can usually be found in a related diagram. A class presented with an empty section at the bottom of the icon indicates that there are no attributes other than those that are visualized using associations.

Certain UML classes are associated with the UML stereotype `<<choice>>`. The `<<choice>>` stereotype specifies that only one of the available properties of the class can be populated at any time. The CybOX UML models utilize `Has_Choice` as the role/property name for associations to `<<choice>>` stereotyped classes. This property is a modeling convention rather than a native element of the underlying data model and acts as a placeholder for one of the available properties of the `<<choice>>` stereotyped class.

1.2.3.1 Class Properties








Generally, a class property can be shown in a UML diagram as either an attribute or an association (i.e., the distinction between attributes and associations is somewhat subjective). In order to make the size of UML diagrams in the specifications manageable, we have chosen to capture most properties as attributes

and to capture only higher level properties as associations, especially in the main top-level component diagrams. In particular, we will always capture properties of UML data types as attributes.

1.2.3.2 Diagram Icons and Arrow Types

Diagram icons are used in a UML diagram to indicate whether a shape is a class, enumeration, or a data type, and decorative icons are used to indicate whether an element is an attribute of a class or an enumeration literal. In addition, two different arrow styles indicate either a directed association relationship (regular arrowhead) or a generalization relationship (triangle-shaped arrowhead). The icons and arrow styles we use are shown and described in [Table 1-1](#).

Table 1-1. UML diagram icons

Icon	Description
	This diagram icon indicates a class. If the name is in italics, it is an abstract class.
	This diagram icon indicates an enumeration.
	This diagram icon indicates a data type.
	This decorator icon indicates an attribute of a class. The green circle means its visibility is public. If the circle is red or yellow, it means its visibility is private or protected.
	This decorator icon indicates an enumeration literal.
	This arrow type indicates a directed association relationship.
	This arrow type indicates a generalization relationship.

1.2.4 Property Table Notation

Throughout Section 3, tables are used to describe the properties of each data model class. Each property table consists of a column of names to identify the property, a type column to reflect the datatype of the property, a multiplicity column to reflect the allowed number of occurrences of the property, and a description column that describes the property. Package prefixes are provided for classes outside of the Network Flow Object data model (see Section 1.2.2).

Note that if a class is a specialization of a superclass, only the properties that constitute the specialization are shown in the property table (i.e., properties of the superclass will not be shown). However, details of the superclass may be shown in the UML diagram.

1.2.5 Property and Class Descriptions

Each class and property defined in CybOX is described using the format, “The X property verb Y.” For example, in the specification for the CybOX Core data model, we write, “The `id` property specifies a globally unique identifier for the Action.” In fact, the verb “specifies” could have been replaced by any number of alternatives: “defines,” “describes,” “contains,” “references,” etc.

However, we thought that using a wide variety of verb phrases might confuse a reader of a specification document because the meaning of each verb could be interpreted slightly differently. On the other hand, we didn't want to use a single, generic verb, such as "describes," because although the different verb choices may or may not be meaningful from an implementation standpoint, a distinction could be useful to those interested in the modeling aspect of CybOX.

Consequently, we have preferred to use the three verbs, defined as follows, in class and property descriptions:

Verb	CybOX Definition
<u>captures</u>	Used to record and preserve information without implying anything about the structure of a class or property. Often used for properties that encompass general content. This is the least precise of the three verbs.
	<p><i>Examples:</i></p> <p>The <code>Observable_Source</code> property characterizes the source of the Observable information. Examples of details <u>captured</u> include identifying characteristics, time-related attributes, and a list of the tools used to collect the information.</p> <p>The <code>Description</code> property <u>captures</u> a textual description of the Action.</p>
<u>characterizes</u>	Describes the distinctive nature or features of a class or property. Often used to describe classes and properties that themselves comprise one or more other properties.
	<p><i>Examples:</i></p> <p>The <code>Action</code> property <u>characterizes</u> a cyber observable Action.</p> <p>The <code>Obfuscation_Technique</code> property <u>characterizes</u> a technique an attacker could potentially leverage to obfuscate the Observable.</p>
<u>specifies</u>	Used to clearly and precisely identify particular instances or values associated with a property. Often used for properties that are defined by a controlled vocabulary or enumeration; typically used for properties that take on only a single value.
	<p><i>Example:</i></p> <p>The <code>cybox_major_version</code> property <u>specifies</u> the major version of the CybOX language used for the set of Observables.</p>

1.3 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [\[RFC2119\]](#).

1.4 Normative References

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

2 Background Information

In this section, we provide high level information about the Network Flow Object data model that is necessary to fully understand the specification details given in Section 3.

2.1 Cyber Observables

A cyber observable is a dynamic event or a stateful property that occurs, or may occur, in the operational cyber domain. Examples of stateful properties include the value of a registry key, the MD5 hash of a file, and an IP address. Examples of events include the deletion of a file, the receipt of an HTTP GET request, and the creation of a remote thread.

A cyber observable is different than a cyber indicator. A cyber observable is a statement of fact, capturing what was observed or could be observed in the cyber operational domain. Cyber indicators are cyber observable patterns, such as a registry key value associated with a known bad actor or a spoofed email address used on a particular date.

2.2 Objects

Cyber observable objects (Files, IP Addresses, etc) in CybOX are characterized with a combination of two levels of data models.

The first level is the Object data model which specifies a base set of properties universal to all types of Objects and enables them to integrate with the overall cyber observable framework specified in the CybOX Core data model.

The second level are the object property models which specify the properties of a particular type of Object via individual data models each focused on a particular cyber entity, such as a Windows registry key, or an Email Message. Accordingly, each release of the CybOX language includes a particular set of Objects that are part of the release. The data model for each of these Objects is defined by its own specification that describes the context-specific classes and properties that compose the Object.

Any specific instance of an Object is represented utilizing the particular object properties data model within the general Object data model.

3 Data Model

3.1 NetworkFlowObjectType Class

The `NetworkFlowObjectType` class specifies the properties necessary to summarize network traffic, expressed as flows of multiple packets. It does not include the packet payload data (i.e. the actual data that was uploaded/downloaded to and from the Dest IP to Source IP as included in packet monitoring tools, such as Wireshark).

The UML diagram corresponding to the `NetworkFlowObjectType` class is shown in [Figure 3-1](#).

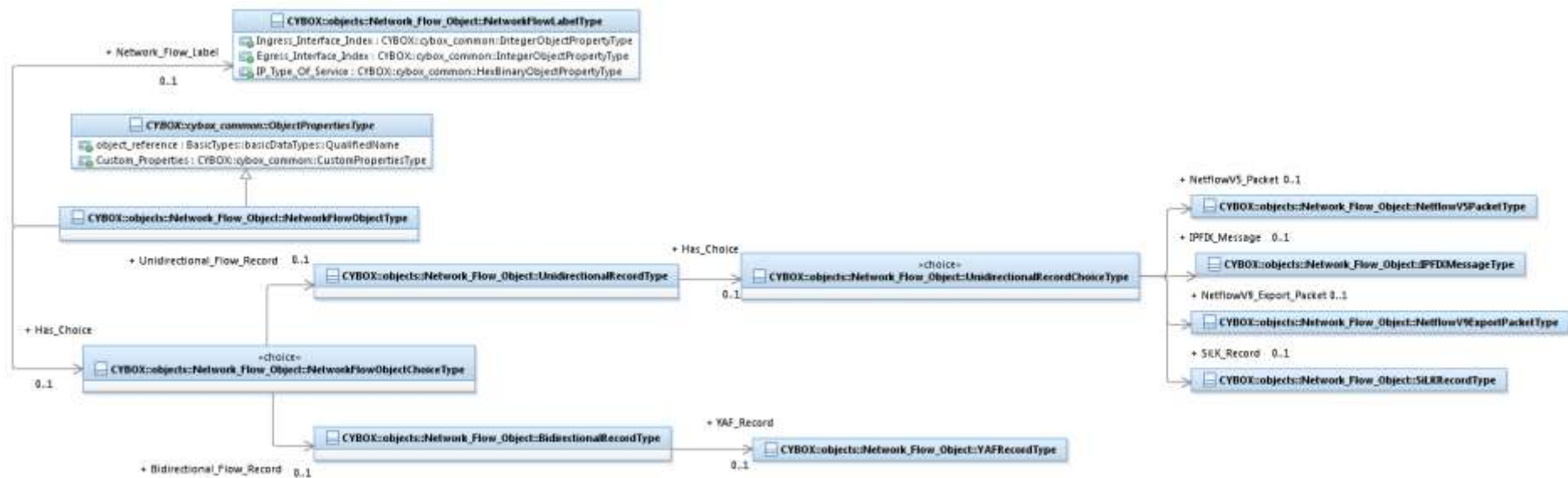


Figure 3-1. UML diagram of the `NetworkFlowObjectType` class

The property table of the `NetworkFlowObjectType` class is given in [Table 3-1](#).

Table 3-1. Properties of the `NetworkFlowObjectType` class

Name	Type	Multiplicity	Description
Network_Flow_Label	NetworkFlowLabelType	0..1	The <code>Network_Flow_Label</code> property represents elements common to all flow records formats - either expressed as a 5-tuple or an extended 7-tuple (actually an 8-tuple because for organizational reasons, we include the egress interface index). Because these properties are defined here, they are excluded from the fields associated directly with each different flow record format type.
Has_Choice	NetworkFlowObjectChoiceType	0..1	<p>The <code>Has_Choice</code> property is associated with the class <code>NetworkFlowObjectChoiceType</code>. It indicates that there is a choice between the <code>Unidirectional_Flow_Record</code> property or the <code>Bidirectional_Flow_Record</code> property.</p> <p>Only one of the properties of <code>NetworkFlowObjectChoiceType</code> class can be populated at any time. See Section 1.2.3 for more detail.</p>

The `NetworkFlowObjectChoiceType` class is the type of the `Has_Choice` property. In the UML model, this class is associated with the <<choice>> UML stereotype, which specifies that only one of the available properties of the `NetworkFlowObjectChoiceType` class can be populated at any time. The property table of the `NetworkFlowObjectChoiceType` class is given in [Table 3-2](#).

Table 3-2. Properties of the `NetworkFlowObjectChoiceType` class

Name	Type	Multiplicity	Description
Unidirectional_Flow_Record	UnidirectionalRecordType	0..1	<p>The <code>Unidirectional_Flow_Record</code> property represents flow-record formats that capture data in one direction only (e.g., Netflow v9).</p> <p>Only one of the <code>Unidirectional_Flow_Record</code> and <code>Bidirectional_Flow_Record</code> properties can be</p>

			populated.
Bidirectional_Flow_Record	BidirectionalRecordType	0..1	<p>The <code>Bidirectional_Flow_Record</code> property represents flow-record formats that capture data in both directions (e.g., YAF).</p> <p>Only one of the <code>Unidirectional_Flow_Record</code> and <code>Bidirectional_Flow_Record</code> properties can be populated.</p>

3.2 NetworkLayerInfoType Class

The `NetworkLayerInfoType` class specifies the network layer information (relative to the OSI network model) which is typically captured in all class of network flow records.

The property table of the `NetworkLayerInfoType` class is given in [Table 3-3](#).

Table 3-3. Properties of the `NetworkLayerInfoType` class

Name	Type	Multiplicity	Description
Src_Socket_Address	SocketAddressObj : SocketAddressObjectType	0..1	The <code>Src_Socket_Address</code> property represents the source IP socket address, consisting of an IP address and port number, for the network flow expressed. Note that not all flow protocols support IPv6 addresses.
Dest_Socket_Address	SocketAddressObj : SocketAddressObjectType	0..1	The <code>Dest_Socket_Address</code> property represents the destination IP socket address, consisting of an IP address and port number, for the network flow expressed. Note that not all flow protocols support IPv6 addresses.
IP_Protocol	PacketObj :	0..1	The <code>IP_Protocol</code> property specifies the IP Protocol of the network flow. This is usually TCP, UDP, or SCTP, but can include

	IANAAssignedIPNumbersType		others as represented in Netflow as an integer from 0 to 255. Please refer to http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xml for reference.
--	---------------------------	--	--

3.3 NetworkFlowLabelType Class

The `NetworkFlowLabelType` class specifies properties that are common to all flow record formats. It builds off of the network layer information (a 5-tuple that commonly defines a flow) and includes ingress and egress interface indexes and IP protocol information (not present in all flow record formats). Egress information is usually not thought of as part of the extended 7-tuple, but we include it for organizational purposes. Because these fields are defined here, they are excluded from the fields associated directly with each different flow record format class.

The property table of the `NetworkFlowLabelType` class is given in [Table 3-4](#).

Table 3-4. Properties of the `NetworkFlowLabelType` class

Name	Type	Multiplicity	Description
Ingress_Interface_Index	cyboxCommon: IntegerObjectPropertyType	0..1	The <code>Ingress_Interface_Index</code> property represents the index (in SNMP, by default) of the network interface card where the flows entered the router.
Egress_Interface_Index	cyboxCommon: IntegerObjectPropertyType	0..1	The <code>Egress_Interface_Index</code> property represents the index (in SNMP, by default) of the network interface card where the flows leave the router.
IP_Type_Of_Service	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>IP_Type_Of_Service</code> property specifies the type of service (ToS) property from the IP header. See http://tools.ietf.org/html/rfc1349.txt for more information.

3.4 UnidirectionalRecordType Class

The `UnidirectionalRecordType` class specifies the netflow record formats that capture traffic in one direction. The UML diagram corresponding to the `UnidirectionalRecordType` class is shown in [Figure 3-1](#).

Name	Type	Multiplicity	Description
Has_Choice	UnidirectionalRecordChoiceType	0..1	<p>The <code>Has_Choice</code> property is associated with the class <code>UnidirectionalRecordChoiceType</code>. It indicates that there is a choice among the various properties .</p> <p>Only one of the properties of <code>UnidirectionalRecordChoiceType</code> class can be populated at any time. See Section 1.2.3 for more detail.</p>

The `UnidirectionalRecordChoiceType` class is the type of the `Has_Choice` property. In the UML model, this class is associated with the <<choice>> UML stereotype, which specifies that only one of the available properties of the `UnidirectionalRecordChoiceType` class can be populated at any time. The property table of the `UnidirectionalRecordChoiceType` class is given in Table 3-5.

Table 3-5. Properties of the `UnidirectionalRecordChoiceType` class

Name	Type	Multiplicity	Description
IPFIX_Message	IPFIXMessageType	0..1	<p>The <code>IPFIX_Message</code> property represents the Internet Protocol Flow Information eXport (IPFIX) protocol. IPFIX is based on Netflow v9. It has several extensions such as Enterprise-defined properties types and variable length fields. See http://tools.ietf.org/html/rfc5101.txt for more information.</p> <p>Only one of the <code>UnidirectionalRecordChoiceType</code> properties can be populated.</p>
NetflowV9_Export_Packet	NetflowV9ExportPacketType	0..1	<p>The <code>NetflowV9_Export_Packet</code> property represents the Netflow V9 flow record format. See https://www.ietf.org/rfc/rfc3954.txt (Netflow v9) for more information.</p>

			Only one of the <code>UnidirectionalRecordChoiceType</code> properties can be populated.
NetflowV5_Packet	<code>NetflowV5PacketType</code>	0..1	<p>The <code>NetflowV5_Packet</code> property represents the Netflow v5 flow record format, which is commonly used to represent network flow data.</p> <p>Only one of the <code>UnidirectionalRecordChoiceType</code> properties can be populated.</p>
SiLK_Record	<code>SiLKRecordType</code>	0..1	<p>The <code>SiLK_Record</code> property represents a network flow record in the System for Internet-Level Knowledge (SiLK) format, developed by CERT at Carnegie Mellon University (CMU)'s Software Engineering Institute (SEI) as part of the NetSA security suite. See http://tools.netsa.cert.org/silk/analysis-handbook.pdf for more information.</p> <p>Only one of the <code>UnidirectionalRecordChoiceType</code> properties can be populated.</p>

3.5 BidirectionalRecordType Class

The `BidirectionalRecordType` class specifies the network record formats that capture traffic in both directions. In the future, we plan to add Argus as a network flow format class. Argus supports bidirectional flows, and as such, is usually used as an alternative to Netflow v5 analysis via SiLK (<http://www.qosient.com/argus/>).

The property table of the `BidirectionalRecordType` class is given in [Table 3-6](#).

Table 3-6. Properties of the `BidirectionalRecordType` class

Name	Type	Multiplicity	Description
------	------	--------------	-------------

YAF_Record	YAFRecordType	0..1	The YAF_Record property represents flow records generated via YAF (Yet Another Flowmeter), a bidirectional network flow meter. See http://www.usenix.org/event/lisa10/tech/full_papers/Inacio.pdf or http://tools.netsa.cert.org/yaf/index.html for more information.
-------------------	---------------	------	---

3.6 IPFIXMessageType Class

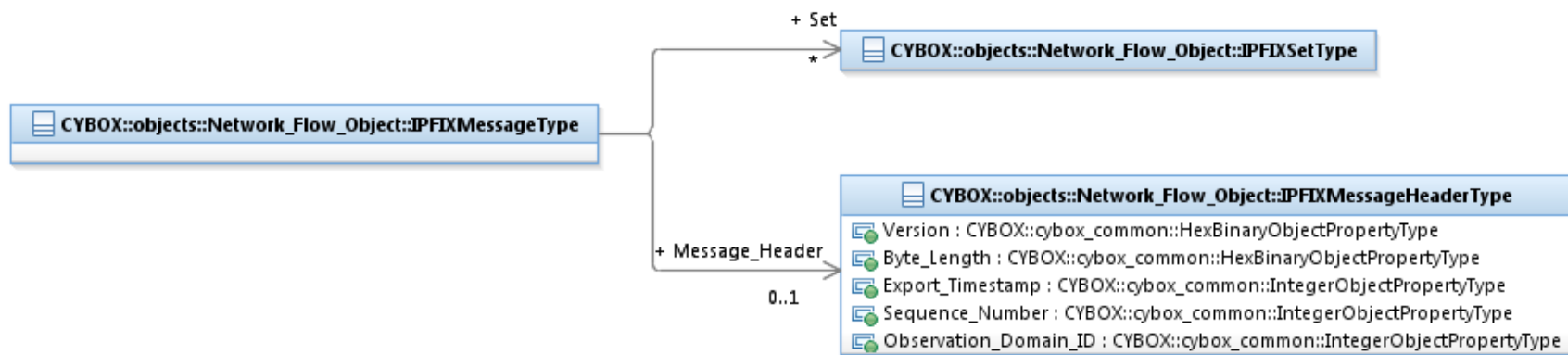


Figure 3-2. UML diagram of the *IPFIXMessageType* class

The *IPFIXMessageType* class specifies the IPFIX protocol which provides IP flow information. See <http://tools.ietf.org/html/rfc5101.txt> for additional information.

The UML diagram corresponding to the *IPFIXMessageType* class is shown in Figure 3-2.

The property table of the *IPFIXMessageType* class is given in Table 3-7.

Table 3-7. Properties of the *IPFIXMessageType* class

Name	Type	Multiplicity	Description
------	------	--------------	-------------

Message_Header	IPFIXMessageHeaderType	0..1	The <code>Message_Header</code> property is the first part of an IPFIX Message, which provides basic information about the message, such as the IPFIX version, length of the message, message sequence number, etc.
Set	IPFIXSetType	0..*	The <code>Set</code> property is a generic term for a collection of records that have a similar structure. In an IPFIX Message, one or more Sets follow the Message Header.

3.6.1 IPFIXMessageHeaderType Class

The `IPFIXMessageHeaderType` class represents the message header for the IPFIX format. For more information about each of the fields, please refer to <http://tools.ietf.org/html/rfc5101.txt> under the heading, "Message Header Field Descriptions." Note that common elements are included in the `Network_Flow_Label`.

The property table of the `IPFIXMessageHeaderType` class is given in [Table 3-8](#).

Table 3-8. Properties of the `IPFIXMessageHeaderType` class

Name	Type	Multiplicity	Description
Version	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>Version</code> property specifies the version number of Flow Record format exported in this message. The value of this property is 0x000a for the current version, incrementing by one the version used in the Netflow services export version 9 (see https://www.ietf.org/rfc/rfc3954.txt).
Byte_Length	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>Byte_Length</code> property indicates the total byte length of the IPFIX Message, measured in octets, including Message Header and Set(s).
Export_Timestamp	cyboxCommon: IntegerObjectPropertyType	0..1	The <code>Export_Timestamp</code> property indicates the time, in seconds, since 0000 UTC Jan 1, 1970, at which the IPFIX message header leaves the Exporter.

Sequence_Number	cyboxCommon: IntegerObjectPropertyType	0..1	The <code>Sequence_Number</code> property indicates the incremental sequence counter modulo 2^{32} of all IPFIX Data Records sent on this PR-SCTP stream from the current Observation Domain by the Exporting Process. This value SHOULD be used by the Collecting Process to identify whether any IPFIX Data Records have been missed. Template and Options Template Records do not increase the Sequence Number.
Observation_Domain_ID	cyboxCommon: IntegerObjectPropertyType	0..1	The <code>Observation_Domain_ID</code> property Indicates a 32-bit identifier of the Observation Domain that is locally unique to the Exporting Process. See http://tools.ietf.org/html/rfc5101.txt under Observation Domain ID for more information.

3.6.2 IPFIXSetType Class

The `IPFIXSetType` class represents the possible sets of records that can be represented in an IPFIX message. See <http://tools.ietf.org/html/rfc5101.txt> under the terms "Template Set", "Options Template Set", and "Data Set", for more information.

The UML diagram corresponding to the `IPFIXMessageType` class is shown in [Figure 3-3](#).

The property table of the `IPFIXSetType` class is given in [Table 3-9](#).

Table 3-9. Properties of the `IPFIXSetType` class

Name	Type	Multiplicity	Description
Has_Choice	<code>IPFIXSetChoiceType</code>	0..1	<p>The <code>Has_Choice</code> property is associated with the class <code>IPFIXSetChoiceType</code>. It indicates that there is a choice among the <code>Template_Set</code>, <code>Options_Template_Set</code>, <code>Data_Set</code> properties.</p> <p>Only one of the properties of <code>IPFIXSetChoiceType</code> class can be populated at any time. See Section 1.2.3 for more</p>

			detail.
--	--	--	---------

The `IPFIXSetChoiceType` class is the type of the `Has_Choice` property. In the UML model, this class is associated with the `<<choice>>` UML stereotype, which specifies that only one of the available properties of the `IPFIXSetChoiceType` class can be populated at any time. The property table of the `IPFIXSetChoiceType` class is given in [Table 3-10](#).

Table 3-10. Properties of the `IPFIXSetChoiceType` class

Name	Type	Multiplicity	Description
Template_Set	<code>IPFIXTemplateSetType</code>	0..1	<p>The <code>Template_Set</code> property indicates a collection of one or more Template Records that have been grouped together in an IPFIX message.</p> <p>Only one of the <code>IPFIXSetChoiceType</code> properties can be populated.</p>
Options_Template_Set	<code>IPFIXOptionsTemplateSetType</code>	0..1	<p>The <code>Options_Template_Set</code> property indicates a collection of one or more Options Template Records that have been grouped together in an IPFIX message.</p> <p>Only one of the <code>IPFIXSetChoiceType</code> properties can be populated.</p>
Data_Set	<code>IPFIXDataSetType</code>	0..1	<p>The <code>Data_Set</code> property indicates one or more Data Records, of the same type, that have been grouped together in an IPFIX message. Each Data Record is previously defined by a Template Record or an Options Template Record.</p> <p>Only one of the <code>IPFIXSetChoiceType</code> properties can be populated.</p>

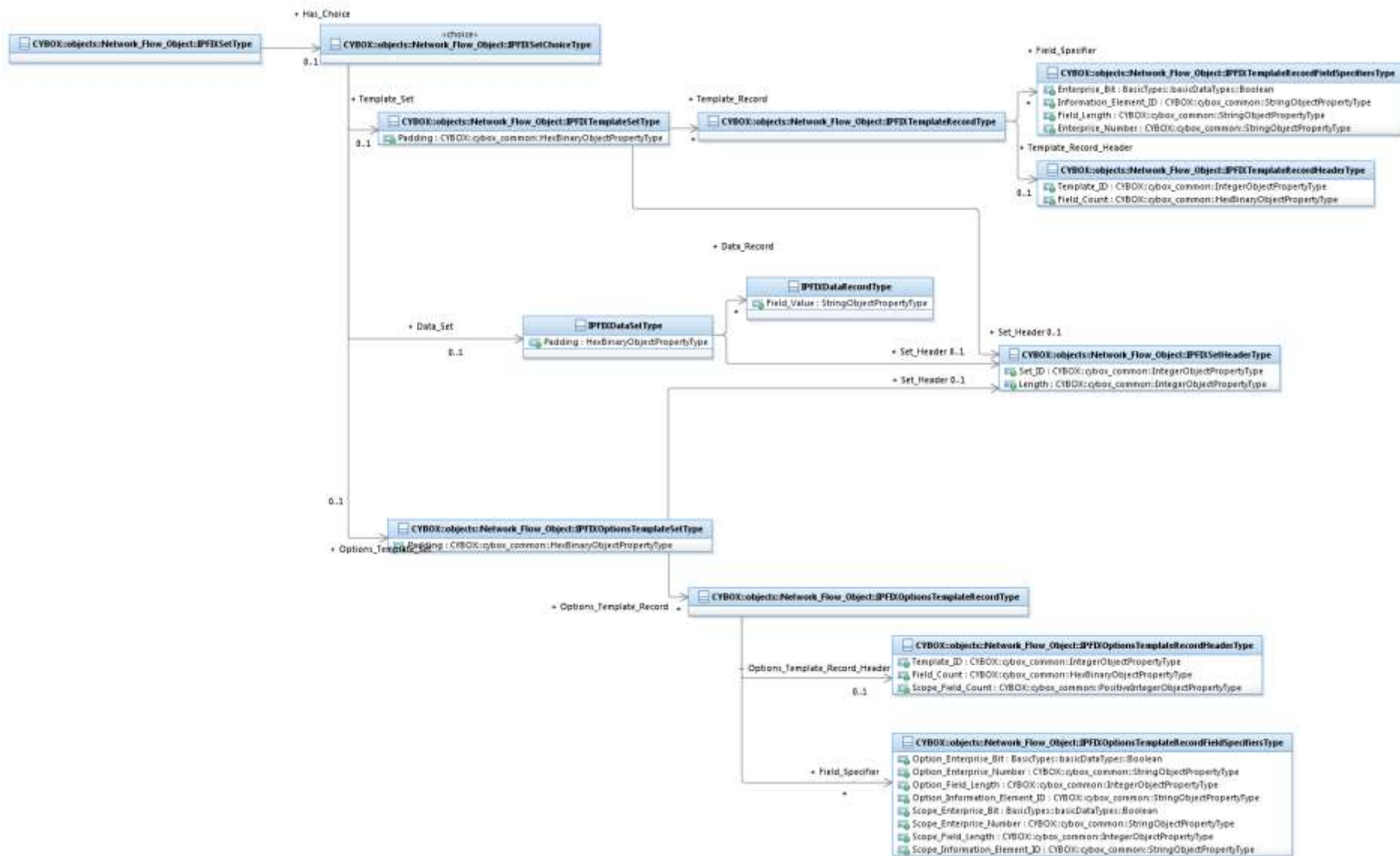


Figure 3-3. UML diagram of the `IPIFXSetType` class

3.6.3 IPFIXTemplateSetType Class

The `IPFIXTemplateSetType` class specifies the regions of a Template Set, of which there are three: the Set Header, the collection of Template Records, and the optional padding at the end of the Template Set. See <http://tools.ietf.org/html/rfc5101.txt> under Set Format, which is section 3.3.1, for more information.

The property table of the `IPFIXTemplateSetType` class is given in [Table 3-11](#).

Table 3-11. Properties of the `IPFIXTemplateSetType` class

Name	Type	Multiplicity	Description
Set_Header	<code>IPFIXSetHeaderType</code>	0..1	The <code>Set_Header</code> property indicates the Set Header region, which is 32-bit region containing the 16-bit properties Set ID and Length.
Template_Record	<code>IPFIXTemplateRecordType</code>	0..*	The <code>Template_Record</code> property indicates the region of Template Records. These are the same properties referenced in the <code>IPFIXTemplateRecordType</code> .
Padding	<code>cyboxCommon:</code> <code>HexBinaryObjectType</code>	0..1	The <code>Padding</code> property indicates the optional Padding at the end of a Template Set. The Exporting Process MAY insert some padding octets, so that the subsequent Set starts at an aligned boundary. For security reasons, the padding octet(s) MUST be composed of zero (0) valued octets, and the padding length MUST be shorter than any allowable record in this Set. For more information, see http://tools.ietf.org/html/rfc5101.txt under Padding.

3.6.4 IPFIXOptionsTemplateSetType Class

The `IPFIXOptionsTemplateSetType` specifies the regions of an Options Template Set, of which there are three: the Set Header, the collection of Options Template Records, and the optional padding at the end of the Options Template Set. See <http://tools.ietf.org/html/rfc5101.txt> under Set Format, which is section 3.3.1, for more information.

The property table of the `IPFIXOptionsTemplateSetType` class is given in [Table 3-12](#).

Table 3-12. Properties of the `IPFIXOptionsTemplateSetType` class

Name	Type	Multiplicity	Description
Set_Header	<code>IPFIXSetHeaderType</code>	0..1	The <code>Set_Header</code> property indicates the Set Header region, which is 32-bit region containing the 16-bit properties Set ID and Length, in that order. These are the same fields referenced in the <code>IPFIXSetHeaderType</code> .
Options_Template_Record	<code>IPFIXOptionsTemplateRecordType</code>	0..*	The <code>Options_Template_Record</code> property indicates the region of Options Template Records. These are the same properties referenced in the <code>IPFIXOptionsTemplateRecordType</code> .
Padding	<code>cyboxCommon:</code> <code>HexBinaryObjectPropertyType</code>	0..1	The <code>Padding</code> property indicates the optional Padding at the end of an Options Template Set. The Exporting Process MAY insert some padding octets, so that the subsequent Set starts at an aligned boundary. For security reasons, the padding octet(s) MUST be composed of zero (0) valued octets, and the padding length MUST be shorter than any allowable record in this Set. For more information, see http://tools.ietf.org/html/rfc5101.txt under Padding.

3.6.5 IPFIXDataSetType Class

The `IPFIXDataSetType` class specifies the regions of a Data Set, of which there are three: the Set Header, the collection of Data Records, and the optional padding at the end of the Data Set. See <http://tools.ietf.org/html/rfc5101.txt> under Set Format, which is section 3.3.1, for more information.

The property table of the `IPFIXDataSetType` class is given in [Table 3-13](#).

Table 3-13. Properties of the `IPFIXDataSetType` class

Name	Type	Multiplicity	Description
Set_Header	IPFIXSetHeaderType	0..1	The <code>Set_Header</code> property indicates the Set Header region, which is 32-bit region containing the 16-bit properties Set ID and Length, appended in that order. These are the same fields referenced in the <code>IPFIXSetHeaderType</code> .
Data_Record	IPFIXDataRecordType	0..*	The <code>Data_Record</code> property indicates the region of Data Records, which consist of a series of property values without a header.
Padding	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>Padding</code> property indicates the optional Padding at the end of a Data Set. The Exporting Process MAY insert some padding octets, so that the subsequent Set starts at an aligned boundary. For security reasons, the padding octet(s) MUST be composed of zero (0) valued octets, and the padding length MUST be shorter than any allowable record in this Set. For more information, see http://tools.ietf.org/html/rfc5101.txt under Padding.

3.6.6 IPFIXSetHeaderType Class

The `IPFIXSetHeaderType` class specifies the properties of the IPFIX set header.

The property table of the `IPFIXSetHeaderType` class is given in [Table 3-14](#).

Table 3-14. Properties of the `IPFIXSetHeaderType` class

Name	Type	Multiplicity	Description
Set_ID	cyboxCommon: IntegerObjectPropertyType	0..1	The <code>Set_ID</code> property Indicates a 16-bit value that identifies the set. The values of 0 and 1 are not used for historical reasons according to https://www.ietf.org/rfc/rfc3954.txt . Otherwise, a value of 2 is reserved for the Template Set and 3 is reserved for the Option Template Set. All other values from 4 to 255 are reserved for future use.

Length	<code>cyboxCommon:</code> <code>IntegerObjectPropertyType</code>	0..1	The <code>Length</code> property Total length of the set, in octets, including the set header, all records, and the optional padding. Because an individual Set MAY contain multiple records, the Length value MUST be used to determine the position of the next Set. See http://tools.ietf.org/html/rfc5101.txt for more information.
---------------	---	------	--

3.6.7 IPFIXTemplateRecordType Class

The `IPFIXTemplateRecordType` class specifies the regions of a Template Record, of which there are two: the Template Record Header, and the Field Specifiers. See <http://tools.ietf.org/html/rfc5101.txt> under Template Record Format, section 3.4.1, for more information.

The property table of the `IPFIXTemplateRecordType` class is given in [Table 3-15](#).

Table 3-15. Properties of the `IPFIXTemplateRecordType` class

Name	Type	Multiplicity	Description
Template_Record_Header	<code>IPFIXTemplateRecordHeaderType</code>	0..1	The <code>Template_Record_Header</code> property indicates the Template Record Header region, which is a 32-bit region containing the 16-bit properties Template ID (> 255) and Field Count, appended in that order. These are the same fields referenced in the <code>IPFIXTemplateRecordHeaderType</code> .
Field_Specifier	<code>IPFIXTemplateRecordFieldSpecifiersType</code>	0..*	The <code>Field_Specifier</code> property indicates the region of Field Specifiers. These are the same properties referenced in the <code>IPFIXTemplateRecordFieldSpecifiersType</code> .

3.6.8 IPFIXTemplateRecordHeaderType Class

The `IPFIXTemplateRecordHeaderType` class specifies the properties in a Template Record Header, `Template_ID` and `Field_Count`, as explained in <http://tools.ietf.org/html/rfc5101.txt>, section 3.4.1.

The property table of the `IPFIXTemplateRecordHeaderType` class is given in [Table 3-16](#).

Table 3-16. Properties of the `IPFIXTemplateRecordHeaderType` class

Name	Type	Multiplicity	Description
Template_ID	<code>cyboxCommon:</code> <code>IntegerObjectPropertyType</code>	0..1	The <code>Template_ID</code> property specifies a unique Template ID which is numbered 256-65535 since IDs 0-255 are reserved for Template Sets, Options Template Sets, and other reserved Sets yet to be created.
Field_Count	<code>cyboxCommon:</code> <code>HexBinaryObjectPropertyType</code>	0..1	The <code>Field_Count</code> property specifies the number of properties in this Template Record.

3.6.9 IPFIXTemplateRecordFieldSpecifiersType Class

The `IPFIXTemplateRecordFieldSpecifiersType` class specifies the fields in a Template Record Field Specifier, as explained in <http://tools.ietf.org/html/rfc5101.txt>, section 3.2.

The property table of the `IPFIXTemplateRecordFieldSpecifiersType` class is given in [Table 3-17](#).

Table 3-17. Properties of the `IPFIXTemplateRecordFieldSpecifiersType` class

Name	Type	Multiplicity	Description
Enterprise_Bit	<code>basicDataTypes:Boolean</code>	0..1	The <code>Enterprise_Bit</code> property specifies the Enterprise bit, either 0 or 1. If this bit is zero, the Information Element Identifier identifies an IETF-specified Information Element, and the four-octet Enterprise Number property SHOULD NOT be present. If this bit is one, the Information Element identifier identifies an enterprise-specific Information Element, and the Enterprise Number field SHOULD be present. NOTE: While it is legal to use "true" and "false" here, this value SHOULD be set to 0 or 1 for consistency.

Information_Element_ID	cyboxCommon: StringObjectPropertyType	0..1	The <code>Information_Element_ID</code> property specifies the 15-bit (NOT 16-bit) Information Element ID referring to the type of Information Element, as shown in https://www.ietf.org/rfc/rfc5102.txt .
Field_Length	cyboxCommon: StringObjectPropertyType	0..1	The <code>Field_Length</code> property specifies the 16-bit Field Length, in octets, of the corresponding encoded Information Element as defined in The property length may be smaller if the reduced size encoding is used (see Section 6.2 of https://www.ietf.org/rfc/rfc5101.txt). The value 65535 is reserved for variable length Information Elements. See https://www.ietf.org/rfc/rfc5102.txt for more information.
Enterprise_Number	cyboxCommon: StringObjectPropertyType	0..1	The <code>Enterprise_Number</code> property specifies the 32-bit IANA Enterprise Number of the authority defining the Information Element identifier in this Template Record. Information Element Identifiers 1.2 and 2.1 are defined by the IETF (Enterprise bit = 0) and, therefore, do not need an Enterprise Number to identify them.

3.6.10 IPFIXOptionsTemplateRecordType Class

The `IPFIXOptionsTemplateRecordType` class specifies the regions of an Options Template Record, of which there are two: the Options Template Record Header, and the Field Specifiers. See <http://tools.ietf.org/html/rfc5101.txt> under Options Template Record Format, section 3.4.2.2, for more information.

The property table of the `IPFIXOptionsTemplateRecordType` class is given in [Table 3-18](#).

Table 3-18. Properties of the `IPFIXOptionsTemplateRecordType` class

Name	Type	Multiplicity	Description
Options_Template_	<code>IPFIXOptionsTemplateRecordHeaderType</code>	0..1	The <code>Options_Template_Record_Header</code> property indicates the Options Template Record

Record_Header			Header region, which is a 48-bit region containing the 16-bit properties Template ID, Field Count, and Scope Field Count, appended in that order.
Field_Specifier	IPFIXOptionsTemplateRecordFieldSpecifiersType	0..*	The <code>Field_Specifier</code> property indicates the region of Field Specifiers. These are the same properties referenced in the <code>IPFIXOptionsTemplateRecordFieldSpecifiersType</code> .

3.6.11 IPFIXOptionsTemplateRecordHeaderType Class

The `IPFIXOptionsTemplateRecordHeaderType` class specifies the header of an options template record.

The property table of the `IPFIXOptionsTemplateRecordHeaderType` class is given in [Table 3-19](#).

Table 3-19. Properties of the `IPFIXOptionsTemplateRecordHeaderType` class

Name	Type	Multiplicity	Description
Template_ID	cyboxCommon: IntegerObjectPropertyType	0..1	The <code>Template_ID</code> property specifies a unique Template ID which is numbered 256-65535 since IDs 0-255 are reserved for Template Sets, Options Template Sets, and other reserved Sets yet to be created.
Field_Count	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>Field_Count</code> property specifies the number of properties in this Options Template Record, INCLUDING the Scope Fields.
Scope_Field_Count	cyboxCommon: PositiveIntegerObjectPropertyType	0..1	The <code>Scope_Field_Count</code> property specifies the number of scope properties in this Options Template Record, which is NONZERO. The Scope Fields are normal Fields except that they are interpreted as scope at the Collector.

3.6.12 IPFIXOptionsTemplateRecordFieldSpecifiersType Class

The `IPFIXOptionsTemplateRecordFieldSpecifiersType` class specifies the properties in an Options Template Record Field Specifier, as explained in <https://www.ietf.org/rfc/rfc5101.txt>, sections 3.2 and 3.4.2.2. It consists of two sequences: Scope Fields and Option Fields, appended together.

The property table of the `IPFIXOptionsTemplateRecordFieldSpecifiersType` class is given in [Table 3-20](#).

Table 3-20. Properties of the `IPFIXOptionsTemplateRecordFieldSpecifiersType` class

Name	Type	Multiplicity	Description
Scope_Enterprise_Bit	<code>basicDataTypes:Boolean</code>	0..1	The <code>Scope_Enterprise_Bit</code> property specifies the Scope Enterprise bit, either 0 or 1. If this bit is zero, the Information Element Identifier identifies an IETF-specified Information Element, and the four-octet Enterprise Number property SHOULD NOT be present. If this bit is one, the Information Element identifier identifies an enterprise-specific Information Element, and the Enterprise Number field SHOULD be present. NOTE: While it is legal to use "true" and "false" here, this value SHOULD be set to 0 or 1.
Scope_Information_Element_ID	<code>cyboxCommon:StringObjectPropertyType</code>	0..1	The <code>Scope_Information_Element_ID</code> property specifies the 15-bit (NOT 16-bit) Scope Information Element ID referring to the type of Information Element. See https://www.ietf.org/rfc/rfc5102.txt for more information.
Scope_Field_Length	<code>cyboxCommon:IntegerObjectPropertyType</code>	0..1	The <code>Scope_Field_Length</code> property specifies the corresponding encoded Information Element, as a 16-bit integer, in octets. The length may be smaller if the reduced size encoding is used (see Section 6.2 of https://www.ietf.org/rfc/rfc5101.txt). The value 65535 is reserved for variable length Information Elements. See https://www.ietf.org/rfc/rfc5102.txt for more information.

Scope_Enterprise_Number	cyboxCommon: StringObjectPropertyType	0..1	The <code>Scope_Enterprise_Number</code> property specifies the 32-bit IANA Scope Enterprise Number of the authority defining the Information Element identifier in this Template Record. Information Element Identifiers 1.2 and 2.1 are defined by the IETF (Enterprise bit = 0) and, therefore, do not need an Enterprise Number to identify them.
Option_Enterprise_Bit	basicDataTypes:Boolean	0..1	The <code>Option_Enterprise_Bit</code> property specifies the Option Enterprise bit, either 0 or 1. If this bit is zero, the Information Element Identifier identifies an IETF-specified Information Element, and the four-octet Enterprise Number property SHOULD NOT be present. If this bit is one, the Information Element identifier identifies an enterprise-specific Information Element, and the Enterprise Number filed SHOULD be present. NOTE: While it is legal to use "true" and "false" here, this value SHOULD be set to 0 or 1 for consistency.
Option_Information_Element_ID	cyboxCommon: StringObjectPropertyType	0..1	The <code>Option_Information_Element_ID</code> property specifies the 15-bit (NOT 16-bit) Option Information Element ID referring to the type of Information Element. See https://www.ietf.org/rfc/rfc5102.txt for more information.
Option_Field_Length	cyboxCommon: IntegerObjectPropertyType	0..1	The <code>Option_Field_Length</code> property specifies the 16-bit Option Field Length, in octets, of the corresponding encoded Information. The property length may be smaller than if the reduced size encoding is used (see Section 6.2 of https://www.ietf.org/rfc/rfc5101.txt). The value 65535 is reserved for variable length Information Elements. See https://www.ietf.org/rfc/rfc5102.txt for more information.
Option_Enterprise_Number	cyboxCommon: StringObjectPropertyType	0..1	The <code>Option_Enterprise_Number</code> property Specifies the 32-bit IANA Option Enterprise Number of the authority defining the Information Element identifier in this Template Record. Information Element Identifiers 1.2 and 2.1 are defined by the IETF (Enterprise bit = 0) and, therefore, do

			not need an Enterprise Number to identify them.
--	--	--	---

3.6.13 IPFIXDataRecordType Class

The `IPFIXDataRecordType` class specifies the data records that are sent in data sets.

The property table of the `IPFIXDataRecordType` class is given in [Table 3-21](#).

Table 3-21. Properties of the `IPFIXDataRecordType` class

Name	Type	Multiplicity	Description
Field_Value	<code>cyboxCommon:</code> <code>StringObjectPropertyType</code>	0..*	The <code>Field_Value</code> property indicates the individual Field Value, which need not be 16-bit. The Template ID to which the Field Values belong to is encoded in the Data Set Header property "Set ID", i.e. "Set ID" = "Template ID".

3.7 NetflowV9ExportPacketType Class

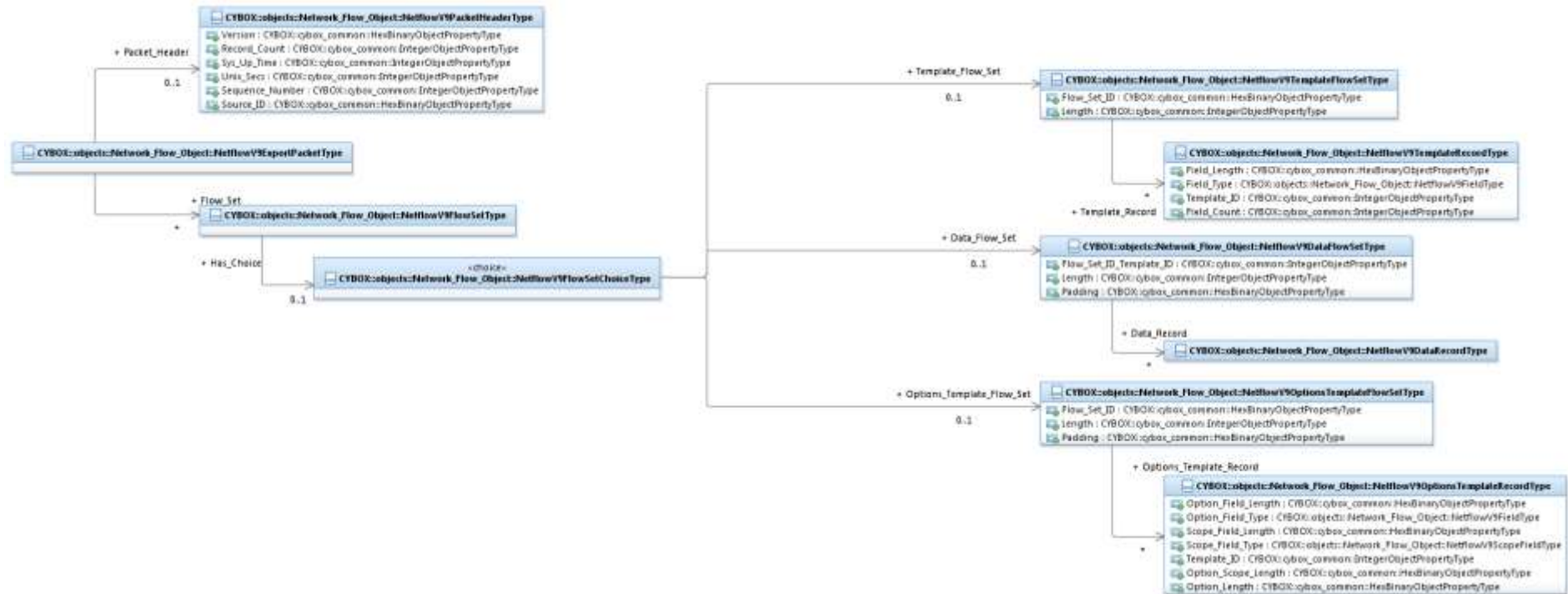


Figure 3-4. UML diagram of the NetworkV9ExportPacketType class

The NetworkFlowObjectType class specifies the Netflow v9 object and specifies the IP flow information. See <http://www.ietf.org/rfc/rfc3954.txt> for more information. It was developed by Cisco.

The UML diagram corresponding to the NetworkFlowObjectType class is shown in Figure 3-4.

The property table of the NetflowV9ExportPacketType class is given in Table 3-22.

Table 3-22. Properties of the NetflowV9ExportPacketType class

Name	Type	Multiplicity	Description
------	------	--------------	-------------

Packet_Header	NetflowV9PacketHeaderType	0..1	The <code>Packet_Header</code> property specifies the Packet Header, which is the first part of an Export Packet. The Packet Header provides basic information about the packet such as the Netflow version, number of records contained within the packet, and sequence numbering.
Flow_Set	NetflowV9FlowSetType	0..*	The <code>Flow_Set</code> property specifies a FlowSet, which is a collection of Flow Records that have similar structure. In an Export Packet, one or more FlowSets follow the Packet Header. There are three different types of FlowSets: a Template FlowSet, Options Template FlowSet, and Data FlowSet.

3.7.1 NetflowV9PacketHeaderType Class

The `NetflowV9PacketHeaderType` class specifies the header properties defined for Netflow v9. Note that common elements are included in the `Network_Flow_Label` property.

See <http://www.ietf.org/rfc/rfc3954.txt> for more information.

The property table of the `NetflowV9PacketHeaderType` class is given in [Table 3-23](#).

Table 3-23. Properties of the `NetflowV9PacketHeaderType` class

Name	Type	Multiplicity	Description
Version	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>Version</code> property specifies the version of flow record format exported in this packet. The value of this property is 9 for the Netflow v9.
Record_Count	cyboxCommon: IntegerObjectPropertyType	0..1	The <code>Record_Count</code> property specifies the total number of records in the Export Packet, which is the sum of Options FlowSet records, Template FlowSet records, and Data FlowSet records. See http://www.ietf.org/rfc/rfc3954.txt .

Sys_Up_Time	cyboxCommon: IntegerObjectPropertyType	0..1	The Sys_Up_Time property specifies the time in milliseconds since this device was first booted.
Unix_Secs	cyboxCommon: IntegerObjectPropertyType	0..1	The Unix_Secs property specifies the time in seconds since 0000 UTC 1970 at which the Export Packet leaves the Exporter.
Sequence_Number	cyboxCommon: IntegerObjectPropertyType	0..1	The Sequence_Number property is an incremental sequence counter of all Export Packets sent from the current Observation Domain by the Exporter. This value MUST be cumulative, and SHOULD be used by the Collector to identify whether any Export Packets have been missed.
Source_ID	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Source_ID property is a 32-bit value that specifies the Exporter Observation Domain. Netflow Collectors SHOULD use the combination of the source IP address and the Source ID property to separate different export streams originating from the same Exporter.

3.7.2 NetflowV9FlowSetType Class

The `NetflowV9FlowSetType` class specifies that one or more in an Export Packet FlowSets follow the Packet Header. There are three different classes of FlowSets, as defined in RFC 3954: Template FlowSet, Options Template FlowSet, and Data FlowSet.

The property table of the `NetflowV9FlowSetType` class is given in [Table 3-24](#).

Table 3-24. Properties of the NetflowV9FlowSetType class

Name	Type	Multiplicity	Description
Has_Choice	NetflowV9FlowSetChoiceType	0..1	The Has_Choice property is associated with the class <code>NetflowV9FlowSetChoiceType</code> . It indicates that there is a choice among the <code>Template_Flow_Set</code> , <code>Options_Template_Flow_Set</code> , and <code>Data_Flow_Set</code>

			<p>properties .</p> <p>Only one of the properties of <code>NetflowV9FlowSetChoiceType</code> class can be populated at any time. See Section 1.2.3 for more detail.</p>
--	--	--	---

The `NetflowV9FlowSetChoiceType` class is the type of the `Has_Choice` property. In the UML model, this class is associated with the `<<choice>>` UML stereotype, which specifies that only one of the available properties of the `NetflowV9FlowSetChoiceType` class can be populated at any time. The property table of the `NetflowV9FlowSetChoiceType` class is given in [Table 3-25](#).

Table 3-25. Properties of the `NetflowV9FlowSetChoiceType` class

Name	Type	Multiplicity	Description
Template_Flow_Set	<code>NetflowV9TemplateFlowSetType</code>	0..1	<p>The <code>Template_Flow_Set</code> property specifies one of the essential elements in the Netflow format is the Template FlowSet. Templates greatly enhance the flexibility of the Flow Record format because they allow the Netflow Collector to process Flow Records without necessarily knowing the interpretation of all the data in the Flow Record.</p> <p>Only one of the <code>NetflowV9FlowSetChoiceType</code> properties can be populated.</p>
Options_Template_Flow_Set	<code>NetflowV9OptionsTemplateFlowSetType</code>	0..1	<p>The <code>Options_Template_Flow_Set</code> property specifies an Options Template FlowSet, which is one or more Options Template Records that have been grouped together in an Export Packet.</p> <p>Only one of the <code>NetflowV9FlowSetChoiceType</code> properties can be populated.</p>

Data_Flow_Set	NetflowV9DataFlowSetType	0..1	<p>The <code>Data_Flow_Set</code> property specifies a Data FlowSet which is one or more records of the same type that are grouped together in an Export Packet. Each record is either a Flow Data Record or an Options Data Record previously defined by a Template Record or an Options Template Record.</p> <p>Only one of the <code>NetflowV9FlowSetChoiceType</code> properties can be populated.</p>
----------------------	--------------------------	------	--

3.7.3 NetflowV9TemplateFlowSetType Class

The `NetflowV9TemplateFlowSetType` class specifies the format of the Template FlowSet.

The property table of the `NetflowV9TemplateFlowSetType` class is given in [Table 3-26](#).

Table 3-26. Properties of the `NetflowV9TemplateFlowSetType` class

Name	Type	Multiplicity	Description
Flow_Set_ID	<code>cyboxCommon:</code> <code>HexBinaryObjectPropertyType</code>	0..1	The <code>Flow_Set_ID</code> property specifies the FlowSet ID, which is fixed to 0 for the Template FlowSet.
Length	<code>cyboxCommon:</code> <code>IntegerObjectPropertyType</code>	0..1	The <code>Length</code> property specifies the sum of the lengths of the FlowSet ID, the Length itself, and all Template Records within this FlowSet.
Template_Record	<code>NetflowV9TemplateRecordType</code>	0..*	The <code>Template_Record</code> property specifies the Template Record region, which includes the template ID, property count, field type, and field length.

3.7.4 NetflowV9TemplateRecordType Class

The `NetflowV9TemplateRecordType` class specifies the Template Record, which includes the template ID, field count, field class, and field length. See <http://www.ietf.org/rfc/rfc3954.txt> for more information.

The property table of the `NetflowV9TemplateRecordType` class is given in [Table 3-27](#).

Table 3-27. Properties of the `NetflowV9TemplateRecordType` class

Name	Type	Multiplicity	Description
Template_ID	<code>cyboxCommon:</code> <code>IntegerObjectPropertyType</code>	0..1	The <code>Template_ID</code> property specifies a unique Template ID for the Template Record. IDs in the range 0-255 are reserved for Template FlowSets, Options FlowSets, and other reserved Sets yet to be created.
Field_Count	<code>cyboxCommon:</code> <code>IntegerObjectPropertyType</code>	0..1	The <code>Field_Count</code> property specifies the number of properties in this Template Record.
Field_Type	<code>NetflowV9FieldType</code>	0..1	The <code>Field_Type</code> property specifies a numeric value that represents the type of the property. Refer to the "Field Type Definitions" section in http://www.ietf.org/rfc/rfc3954.txt for descriptions of these types.
Field_Length	<code>cyboxCommon:</code> <code>HexBinaryObjectPropertyType</code>	0..1	The <code>Field_Length</code> property specifies the length of the corresponding property type, in bytes.

3.7.5 NetflowV9FieldType Data Type

The `NetflowV9FieldType` data type specifies the field. Its core value SHOULD be a literal found in the `NetflowV9FieldTypeEnum` enumeration. Its base type is the `BaseObjectPropertyType` data type, in order to permit complex (i.e. regular-expression based) specifications.

3.7.6 NetflowV9OptionsTemplateFlowSetType Class

The `NetflowV9OptionsTemplateFlowSetType` class specifies an Options Template FlowSet, which is one or more Options Template Records that have been grouped together in an Export Packet.

The property table of the `NetflowV9OptionsTemplateFlowSetType` class is given in [Table 3-28](#).

Table 3-28. Properties of the `NetflowV9OptionsTemplateFlowSetType` class

Name	Type	Multiplicity	Description
Flow_Set_ID	<code>cyboxCommon:</code> <code>HexBinaryObjectPropertyType</code>	0..1	The <code>Flow_Set_ID</code> property specifies the FlowSet ID, which is fixed to 1 for the Options Template FlowSet.
Length	<code>cyboxCommon:</code> <code>IntegerObjectPropertyType</code>	0..1	The <code>Length</code> property specifies the total length of this FlowSet in octets, including the set header, all records, and the optional padding.
Options_Template_Record	<code>NetflowV9OptionsTemplateRecordType</code>	0..*	The <code>Options_Template_Record</code> property specifies the Options Template Record region which includes the Option Scope Length, Option Length, and properties specifying the Scope field type and Scope field length.
Padding	<code>cyboxCommon:</code> <code>HexBinaryObjectPropertyType</code>	0..1	The <code>Padding</code> property specifies the number of padding bytes to be inserted so that the subsequent FlowSet starts at a 4-byte aligned boundary. It is important to note that the <code>Length</code> property includes the padding bytes. Padding SHOULD be using zeros.

3.7.7 NetflowV9OptionsTemplateRecordType Class

The `NetflowV9OptionsTemplateRecordType` class specifies the Options Template Record which includes the Option Scope Length, Option Length, and fields specifying the Scope field class and Scope field length.

The property table of the `NetflowV9OptionsTemplateRecordType` class is given in [Table 3-29](#).

Table 3-29. Properties of the `NetflowV9OptionsTemplateRecordType` class

Name	Type	Multiplicity	Description
Template_ID	cyboxCommon: IntegerObjectPropertyType	0..1	The <code>Template_ID</code> property specifies the template ID of this Options Template, which must be greater than 255.
Option_Scope_Length	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>Option_Scope_Length</code> property specifies the length of bytes of any <code>Scope</code> property definition contained in the Options Template Record.
Option_Length	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>Option_Length</code> property specifies the length of bytes of any options property definitions contained in this Options Template Record.
Scope_Field_Type	NetflowV9ScopeFieldType	0..1	The <code>Scope_Field_Type</code> property specifies the relevant portion of the Exporter/Netflow process to which the Options Template Record refers. Currently defined values include 1 for System, 2 for Interface, 3 for Line Card, 4 for Cache, and 5 for Template. See http://www.ietf.org/rfc/rfc3954.txt for more information.
Scope_Field_Length	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>Scope_Field_Length</code> property specifies the length (in bytes) of the <code>Scope</code> property as it would appear in an Options Data Record.
Option_Field_Type	NetflowV9FieldType	0..1	The <code>Option_Field_Type</code> property specifies the type of property that would appear in the Options Template Record. See http://www.ietf.org/rfc/rfc3954.txt for more information.
Option_Field_Length	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>Option_Field_Length</code> property specifies the length (in bytes) of the <code>Option</code> property.

3.7.8 NetflowV9ScopeFieldType Data Type

The `NetflowV9ScopeFieldType` data type specifies the scope field. Its core value SHOULD be a literal found in the `NetflowV9ScopeFieldTypeEnum` enumeration. Its base type is the `BaseObjectPropertyType` data type, in order to permit complex (i.e. regular-expression based) specifications.

3.7.9 NetflowV9DataFlowSetType Class

The `NetflowV9DataFlowSetType` class specifies a Data FlowSet, which is one or more records of the same class that are grouped together in an Export Packet. Each record is either a Flow Data Record or an Options Data Record previously defined by a Template Record or an Options Template Record. See <http://www.ietf.org/rfc/rfc3954.txt> for more information.

The property table of the `NetflowV9DataFlowSetType` class is given in [Table 3-30](#).

Table 3-30. Properties of the `NetflowV9DataFlowSetType` class

Name	Type	Multiplicity	Description
Flow_Set_ID_Template_ID	<code>cyboxCommon:</code> <code>IntegerObjectPropertyType</code>	0..1	The <code>Flow_Set_ID_Template_ID</code> property specifies the FlowSet ID, which corresponds to the Template ID from a Template Flow Set or an Options Template Flow Set.
Length	<code>cyboxCommon:</code> <code>IntegerObjectPropertyType</code>	0..1	The <code>Length</code> property specifies the length of this FlowSet.
Data_Record	<code>NetflowV9DataRecordType</code>	0..*	The <code>Data_Record</code> property contains a collection of Flow Data Record(s), each containing a set of property values. The Type and Length of the fields have been previously defined in the Template Record referenced by the FlowSet ID or Template ID. The data record specifies either a template flow set or an options template flow set.
Padding	<code>cyboxCommon:</code> <code>HexBinaryObjectPropertyType</code>	0..1	The <code>Padding</code> property specifies the padding bytes used so that the subsequent FlowSet starts at a 4-byte aligned

			boundary. It is important to note that the Length property includes the padding bytes. Padding SHOULD be using zeros.
--	--	--	---

3.7.10 NetflowV9DataRecordType Class

The `NetflowV9DataRecordType` class specifies a Data FlowSet, which is one or more records of the same class that are grouped together in an Export Packet. Each record is either a Flow Data Record or an Options Data Record previously defined by a Template Record or an Options Template Record. See <http://www.ietf.org/rfc/rfc3954.txt>. The UML diagram corresponding to the `NetworkFlowObjectType` class is shown in Figure 3-5.

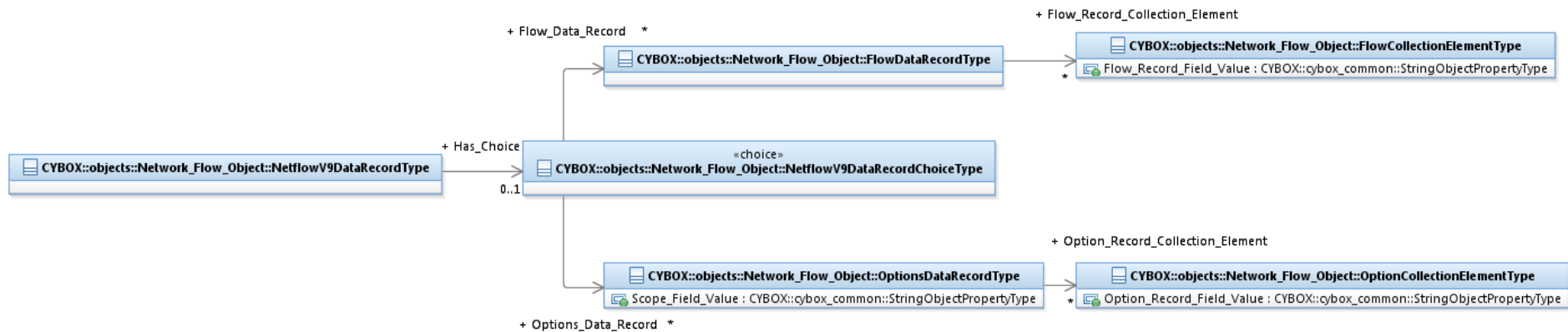


Figure 3-5. UML diagram of the `NetflowV9DataRecordType` class

The property table of the `NetflowV9DataRecordType` class is given in Table 3-31.

Table 3-31. Properties of the `NetflowV9DataRecordType` class

Name	Type	Multiplicity	Description
Has_Choice	NetflowV9DataRecordChoiceType	0..1	The Has_Choice property is associated with the class

			<p>NetflowV9DataRecordChoiceType. It indicates that there is a choice between the Flow_Data_Record property and the Options_Data_Record property.</p> <p>Only one of the properties of NetflowV9DataRecordChoiceType class can be populated at any time. See Section 1.2.3 for more detail.</p>
--	--	--	---

The NetflowV9DataRecordChoiceType class is the type of the Has_Choice property. In the UML model, this class is associated with the <<choice>> UML stereotype, which specifies that only one of the available properties of the NetflowV9DataRecordChoiceType class can be populated at any time. The property table of the NetflowV9DataRecordChoiceType class is given in [Table 3-32](#).

Table 3-32. Properties of the NetflowV9DataRecordChoiceType class

Name	Type	Multiplicity	Description
Flow_Data_Record	FlowDataRecordType	0..*	<p>The Flow_Data_Record property specifies a Flow Data Record, which corresponds to a FieldType defined in the Template Record. Each one will have multiple values associated with it.</p> <p>The Flow_Data_Record and Options_Data_Record properties MUST NOT both have a value.</p>
Options_Data_Record	OptionsDataRecordType	0..*	<p>The Options_Data_Record property specifies an Options Data Record, which corresponds to a previously defined Options Template Record.</p> <p>The Flow_Data_Record and Options_Data_Record properties MUST NOT both have a value.</p>

3.7.11 FlowDataRecordType Class

The `FlowDataRecordType` class specifies a data record that contains values of the Flow parameters corresponding to a Template Record.

The property table of the `FlowDataRecordType` class is given in [Table 3-33](#).

Table 3-33. Properties of the `FlowDataRecordType` class

Name	Type	Multiplicity	Description
Flow_Record_Collection_Element	<code>FlowCollectionElementType</code>	0..*	The <code>Flow_Record_Collection_Element</code> property specifies property values for each flow record.

3.7.12 FlowCollectionElementType Class

The `FlowCollectionElementType` class specifies the values that are associated with each record in the collection of a flow data record.

The property table of the `FlowCollectionElementType` class is given in [Table 3-34](#).

Table 3-34. Properties of the `FlowCollectionElementType` class

Name	Type	Multiplicity	Description
Flow_Record_Field_Value	<code>cyboxCommon:</code> <code>StringObjectPropertyType</code>	0..*	The <code>Flow_Record_Field_Value</code> property specifies the set of property values for a given Flow Data Record.

3.7.13 OptionsDataRecordType Class

The `OptionsDataRecordType` class specifies the data record that contains values and scope information of the Flow measurement parameters, corresponding to an Options Template Record.

The property table of the `OptionsDataRecordType` class is given in [Table 3-35](#).

Table 3-35. Properties of the OptionsDataRecordType class

Name	Type	Multiplicity	Description
Scope_Field_Value	cyboxCommon: StringObjectPropertyType	0..1	The Scope_Field_Value property corresponds to a previously defined Options Template Record.
Option_Record_Collection_Element	OptionCollectionElementType	0..*	The Option_Record_Collection_Element property specifies property values for each flow record.

3.7.14 OptionCollectionElementType Class

The OptionCollectionElementType class specifies the property values that are associated with each option in the collection of an option data record.

The property table of the OptionCollectionElementType class is given in [Table 3-36](#).

Table 3-36. Properties of the OptionCollectionElementType class

Name	Type	Multiplicity	Description
Option_Record_Field_Value	cyboxCommon: StringObjectPropertyType	0..*	The Option_Record_Field_Value property specifies the set of property values for a given Option Data Record.

3.8 NetflowV5PacketType Class

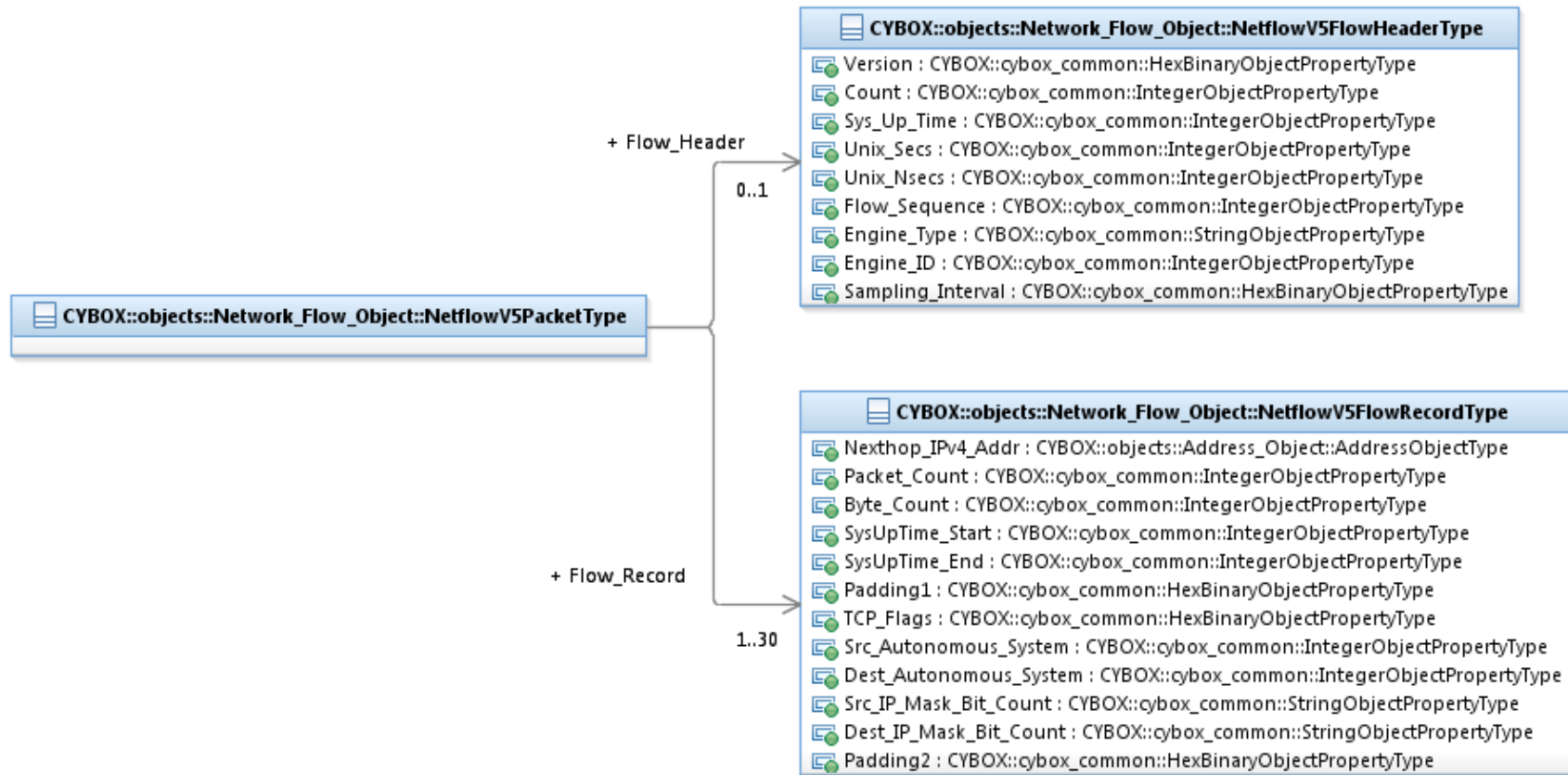


Figure 3-6. UML diagram of the *NetworkV5PacketType* class

The *NetflowV5PacketType* class specifies the contents of a Netflow v5 packet. As of 2012, Netflow v5 is still the most commonly used network flow format. Netflow v5 was developed by Cisco. See http://netflow.caligare.com/netflow_v5.htm for more information.

The UML diagram corresponding to the *NetworkFlowObjectType* class is shown in **Figure 3-6**.

The property table of the *NetflowV5PacketType* class is given in **Table 3-37**.

Table 3-37. Properties of the `NetflowV5PacketType` class

Name	Type	Multiplicity	Description
Flow_Header	<code>NetflowV5FlowHeaderType</code>	0..1	The <code>Flow_Header</code> property specifies properties of a Netflow v5 header.
Flow_Record	<code>NetflowV5FlowRecordType</code>	1..30	The <code>Flow_Record</code> property specifies the elements of a Netflow v5 flow record. See http://netflow.caligare.com/netflow_v5.htm or http://tools.netsa.cert.org/silk/faq.html#ipfix-fields for more information.

3.8.1 NetflowV5FlowHeaderType Class

The `NetflowV5FlowHeaderType` class specifies properties of a Netflow v5 header. See http://netflow.caligare.com/netflow_v5.htm for more information.

The property table of the `NetflowV5FlowHeaderType` class is given in [Table 3-38](#).

Table 3-38. Properties of the `NetflowV5FlowHeaderType` class

Name	Type	Multiplicity	Description
Version	<code>cyboxCommon:</code> <code>HexBinaryObjectPropertyType</code>	0..1	The <code>Version</code> property specifies the Netflow export format version number, which defaults to 5 in this case.
Count	<code>cyboxCommon:</code> <code>IntegerObjectPropertyType</code>	0..1	The <code>Count</code> property specifies the number of flows exported in the packet (1-30).
Sys_Up_Time	<code>cyboxCommon:</code> <code>IntegerObjectPropertyType</code>	0..1	The <code>Sys_Up_Time</code> property specifies the current time in milliseconds since the export device booted.
Unix_Secs	<code>cyboxCommon:</code>	0..1	The <code>Unix_Secs</code> property specifies the current time in milliseconds

	IntegerObjectPropertyType		since 0000 UTC 1970.
Unix_Nsecs	cyboxCommon: IntegerObjectPropertyType	0..1	The <code>Unix_Nsecs</code> property specifies the residual in nanoseconds since 0000 UTC 1970.
Flow_Sequence	cyboxCommon: IntegerObjectPropertyType	0..1	The <code>Flow_Sequence</code> property specifies the sequence counter of total flows seen.
Engine_Type	cyboxCommon: StringObjectPropertyType	0..1	The <code>Engine_Type</code> property specifies the type of flow-switching engine.
Engine_ID	cyboxCommon: IntegerObjectPropertyType	0..1	The <code>Engine_ID</code> property specifies the slot number of the flow-switching engine.
Sampling_Interval	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>Sampling_Interval</code> property specifies the first two bits holding the sampling mode, with the remaining 14 bits holding the value of the sampling interval.

3.8.2 NetflowV5FlowRecordType Class

The `NetflowV5FlowRecordType` class specifies properties of a Netflow v5 flow record. Recall that the seven elements that define the flow itself (e.g., source IP address) are provided in `NetworkFlowLabelType`. See <https://bto.bluecoat.com/packetguide/8.6/info/netflow5-records.htm> for more information.

The property table of the `NetflowV5FlowRecordType` class is given in [Table 3-39](#).

Table 3-39. Properties of the `NetflowV5FlowRecordType` class

Name	Type	Multiplicity	Description
Nexthop_IPv4_Addr	AddressObj :	0..1	The <code>Nexthop_IPv4_Addr</code> property represents the IP

	AddressObjectType		address of the next hop router.
Packet_Count	cyboxCommon: IntegerObjectPropertyType	0..1	The Packet_Count property represents the number of packets in the flow.
Byte_Count	cyboxCommon: IntegerObjectPropertyType	0..1	The Byte_Count property represents the total number of bytes in the flow.
SysUpTime_Start	cyboxCommon: IntegerObjectPropertyType	0..1	The SysUpTime_Start property represents the SysUpTime at start of flow: the total time in milliseconds starting from when the first packet in the flow was seen.
SysUpTime_End	cyboxCommon: IntegerObjectPropertyType	0..1	The SysUpTime_End property represents the SysUpTime at end of flow: when the last packet in the flow was seen.
Padding1	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Padding1 property specifies one byte of padding.
TCP_Flags	cyboxCommon: HexBinaryObjectPropertyType	0..1	The TCP_Flags property specifies the union of all TCP flags observed over the life of the flow.
Src_Autonomous_System	cyboxCommon: IntegerObjectPropertyType	0..1	The Src_Autonomous_System property specifies the source autonomous system number, either origin or peer.
Dest_Autonomous_System	cyboxCommon: IntegerObjectPropertyType	0..1	The Dest_Autonomous_System property specifies the destination autonomous system number, either origin or peer.
Src_IP_Mask_Bit_Count	cyboxCommon: StringObjectPropertyType	0..1	The Src_IP_Mask_Bit_Count property specifies the source address prefix mask bits.

Dest_IP_Mask_Bit_Count	cyboxCommon: StringObjectPropertyType	0..1	The Dest_IP_Mask_Bit_Count property specifies the destination address prefix mask bits.
Padding2	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Padding2 property specifies the unused (zero) bytes, which is used for purposes of padding.

3.9 SiLKRecordType Class

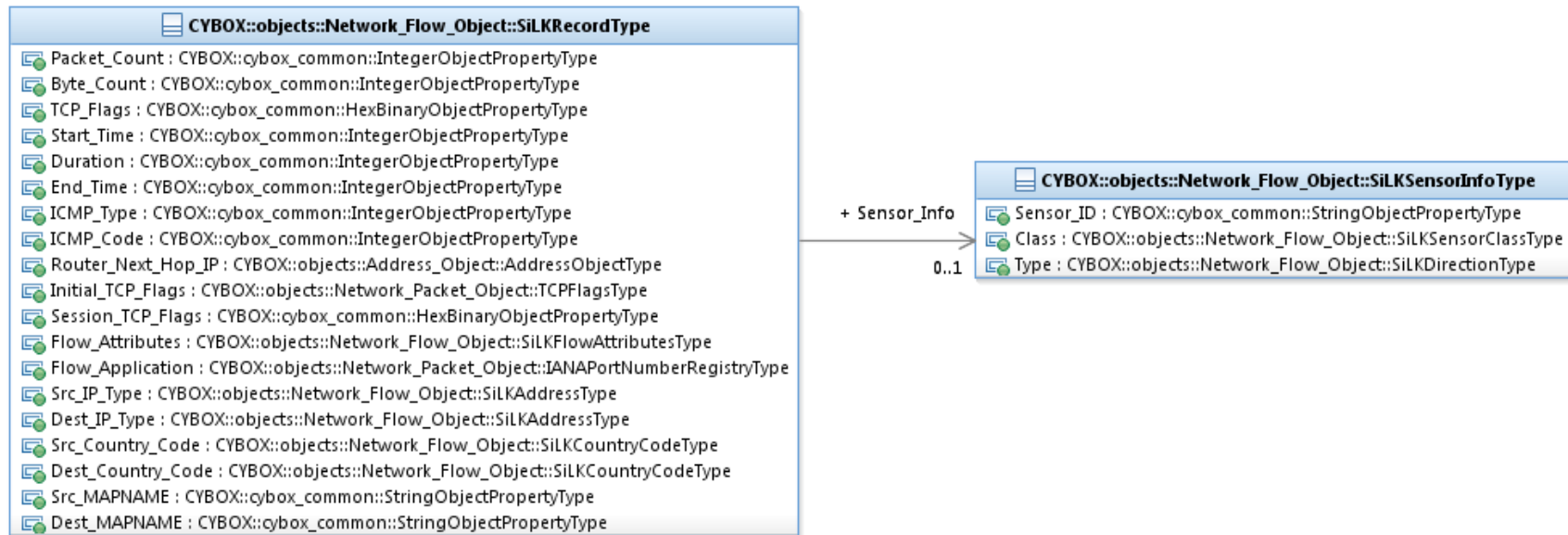


Figure 3-7. UML diagram of the *SiLKRecordType* class

The *SiLKRecordType* class, specifies the System for Internet-Level Knowledge (CMU/SEI) record type. The properties are taken from a list shown in <http://tools.netsa.cert.org/silk/rwcut.html>. Fields common to all network flows are defined in *NetworkFlowLabelType* class (e.g., source IP, SNMP ingress, etc.). For additional references, see <http://tools.netsa.cert.org/silk/analysis-handbook.pdf> and <http://tools.netsa.cert.org/silk/faq.html#ipfix-fields>.

The UML diagram corresponding to the *NetworkFlowObjectType* class is shown in **Figure 3-7**.

The property table of the `SiLKRecordType` class is given in [Table 3-40](#).

Table 3-40. Properties of the `SiLKRecordType` class

Name	Type	Multiplicity	Description
Packet_Count	<code>cyboxCommon:</code> <code>IntegerObjectPropertyType</code>	0..1	The <code>Packet_Count</code> property represents the number of packets in the flow.
Byte_Count	<code>cyboxCommon:</code> <code>IntegerObjectPropertyType</code>	0..1	The <code>Byte_Count</code> property represents the number of Layer 3 bytes in the packets of the flow.
TCP_Flags	<code>cyboxCommon:</code> <code>HexBinaryObjectPropertyType</code>	0..1	The <code>TCP_Flags</code> property specifies the union of all TCP flags observed over the life of the flow.
Start_Time	<code>cyboxCommon:</code> <code>IntegerObjectPropertyType</code>	0..1	The <code>Start_Time</code> property represents the <code>SysUpTime</code> at start of flow, i.e. the total time in milliseconds starting from when the router booted. There is another element " <code>Start_Time + msec</code> " which is the starting time of flow including milliseconds, but milliseconds are the resolution of <code>Start_Time</code> unless the <code>-legacy-timestamps</code> switch is specified, so " <code>Start_Time + msec</code> " is not defined separately.
Duration	<code>cyboxCommon:</code> <code>IntegerObjectPropertyType</code>	0..1	The <code>Duration</code> property specifies the duration of the flow. There is another element " <code>Duration + msec</code> " which is the starting time of flow including milliseconds, but milliseconds are the resolution of <code>Duration</code> unless the <code>-legacy-timestamps</code> switch is specified, so " <code>Duration + msec</code> " is not defined separately.
End_Time	<code>cyboxCommon:</code> <code>IntegerObjectPropertyType</code>	0..1	The <code>End_Time</code> property represents the <code>SysUpTime</code> at end of flow. There is another element " <code>End_Time + msec</code> " which is the starting time of flow including milliseconds, but milliseconds are the resolution of <code>End_Time</code> unless the <code>-legacy-timestamps</code> switch is specified, so " <code>End_Time + msec</code> " is not defined

			separately.
Sensor_Info	SiLKSensorInfoType	0..1	The <code>Sensor_Info</code> property defines the properties associated with the sensor at the collection point.
ICMP_Type	cyboxCommon: IntegerObjectPropertyType	0..1	The <code>ICMP_Type</code> property specifies the type for ICMP flows. It is empty for non-ICMP flows.
ICMP_Code	cyboxCommon: IntegerObjectPropertyType	0..1	The <code>ICMP_Code</code> property specifies the code for ICMP flows. It is empty for non-ICMP flows.
Router_Next_Hop_IP	AddressObj: AddressObjectType	0..1	The <code>Router_Next_Hop_IP</code> property specifies the router next hop IP.
Initial_TCP_Flags	PacketObj:TCPFlagsType	0..1	The <code>Initial_TCP_Flags</code> property specifies the TCP flags on first packet in the flow.
Session_TCP_Flags	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>Session_TCP_Flags</code> property specifies the bit-wise OR of TCP flags over all packets except the first in the flow.
Flow_Attributes	SiLKFlowAttributesType	0..1	The <code>Flow_Attributes</code> property specifies the flow attributes set by the flow generator.
Flow_Application	PacketObj: IANAPortNumberRegistryType	0..1	The <code>Flow_Application</code> property is based on an examination of payload contents. The value is equal to the port number traditionally used for that type of traffic (21 for FTP traffic even if actually routed over port 80). Documentation (http://tools.netsa.cert.org/silk/rwcut.html) says this is a "guess as to the content of the flow".
Src_IP_Type	SiLKAddressType	0..1	The <code>Src_IP_Type</code> property specifies the type of the source IP in terms of whether the address is routable, external, etc.

Dest_IP_Type	SiLKAddressType	0..1	The <code>Dest_IP_Type</code> property specifies the type of the destination IP in terms of whether the address is routable, external, etc.
Src_Country_Code	SiLKCOUNTRYCodeType	0..1	The <code>Src_Country_Code</code> property specifies a two-letter country code denoting the country of location of the source IP address.
Dest_Country_Code	SiLKCOUNTRYCodeType	0..1	The <code>Dest_Country_Code</code> property specifies a two-letter country code denoting the country of location of the destination IP address.
Src_MAPNAME	cyboxCommon: StringObjectPropertyType	0..1	The <code>Src_MAPNAME</code> property specifies the user defined string for integrating external information into SiLK records. See documentation on SiLK pmap filter for details (defined in the prefix map associated with MAPNAME).
Dest_MAPNAME	cyboxCommon: StringObjectPropertyType	0..1	The <code>Dest_MAPNAME</code> property specifies the user defined string for integrating external information into SiLK records. See documentation on SiLK pmap filter for details (defined in the prefix map associated with MAPNAME).

3.9.1 SiLKFlowAttributesType Data Type

The `SiLKFlowAttributesType` data type specifies the SiLK flow attributes. Its core value SHOULD be a literal found in the `SiLKFlowAttributesTypeEnum` enumeration. Its base type is the `BaseObjectPropertyType` data type, in order to permit complex (i.e. regular-expression based) specifications.

3.9.2 SiLKAddressType Data Type

The `SiLKAddressType` data type specifies the SiLK address type. Its core value SHOULD be a literal found in the `SiLKAddressTypeEnum` enumeration. Its base type is the `BaseObjectPropertyType` data type, in order to permit complex (i.e. regular-expression based) specifications.

3.9.3 SiLKCOUNTRYCODEType Class

The `SiLKCOUNTRYCODEType` data type specifies the country codes used. Its core value SHOULD be a literal found in the `SiLKCOUNTRYCODETypeEnum` enumeration. Its base type is the `BaseObjectPropertyType` data type, in order to permit complex (i.e. regular-expression based) specifications.

3.9.4 SiLKSensorInfoType Class

The `SiLKSensorInfoType` class specifies properties associated with a SiLK sensor.

The property table of the `SiLKSensorInfoType` class is given in [Table 3-41](#).

Table 3-41. Properties of the `SiLKSensorInfoType` class

Name	Type	Multiplicity	Description
Sensor_ID	<code>cyboxCommon:StringObjectPropertyType</code>	0..1	The <code>Sensor_ID</code> property specifies the name or ID of sensor at the collection point.
Class	<code>SiLKSensorClassType</code>	0..1	The <code>Class</code> property specifies the sensor class. By default, the "all" class. Others can be configured.
Type	<code>SiLKDirectionType</code>	0..1	The <code>Type</code> property specifies the direction of traffic, which is enumerated by <code>SiLKDirectionType</code> .

3.9.5 SiLKDirectionType Class

The `SiLKDirectionType` data type specifies the direction of SiLK traffic. Its core value SHOULD be a literal found in the `SiLKDirectionTypeEnum` enumeration. Its base type is the `BaseObjectPropertyType` data type, in order to permit complex (i.e. regular-expression based) specifications.

3.9.6 SiLKSensorClassType Class

The `SiLKSensorClassType` data type specifies the sensor type. Its core value SHOULD be a literal found in the `SiLKSensorClassTypeEnum` enumeration. Its base type is the `BaseObjectPropertyType` data type, in order to permit complex (i.e. regular-expression based) specifications.

3.10 YAFRecordType Class

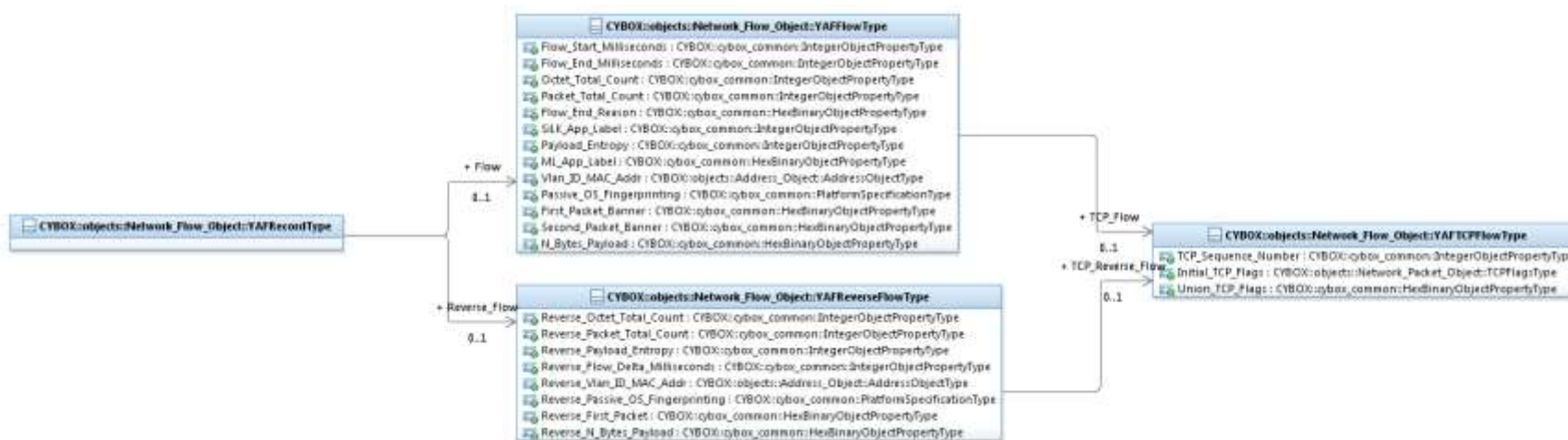


Figure 3-8. UML diagram of the `YAFRecordType` class

The `YAFRecordType` class specifies the YAF (Yet Another Flowmeter) record type, which is a bidirectional network flow meter. It processes packet data from pcap(3) dumpfiles as generated by tcpdump(1) or via live capture from an interface using pcap(3) into bidirectional flows, then exports those flows to IPFIX. See http://www.usenix.org/event/lisa10/tech/full_papers/Inacio.pdf for more information.

The UML diagram corresponding to the `NetworkFlowObjectType` class is shown in [Figure 3-8](#).

The property table of the `YAFRecordType` class is given in [Table 3-42](#).

Table 3-42. Properties of the `YAFRecordType` class

Name	Type	Multiplicity	Description
------	------	--------------	-------------

Flow	YAFFlowType	0..1	The <code>Flow</code> property specifies the properties in a YAF record that have been separated based on flow direction. These properties are defined for the general forward flow.
Reverse_Flow	YAFReverseFlowType	0..1	The <code>Reverse_Flow</code> property specifies some of the properties in a YAF record which correspond to the reverse flow.

3.10.1 YAFFlowType Class

The `YAFFlowType` class specifies the properties of a YAF record correspond to the flow generally or to the forward portion of the flow. Properties common to all network flow objects are defined in the `NetworkFlowLabelType` class (src ip address, ingress/egress interface).

The property table of the `YAFFlowType` class is given in [Table 3-43](#).

Table 3-43. Properties of the `YAFFlowType` class

Name	Type	Multiplicity	Description
Flow_Start_Milliseconds	cyboxCommon: IntegerObjectPropertyType	0..1	The <code>Flow_Start_Milliseconds</code> property specifies the flow start time in milliseconds since 1970-01-01 00:00:00 UTC.
Flow_End_Milliseconds	cyboxCommon: IntegerObjectPropertyType	0..1	The <code>Flow_End_Milliseconds</code> property specifies the flow end time in milliseconds since 1970-01-01 00:00:00 UTC.
Octet_Total_Count	cyboxCommon: IntegerObjectPropertyType	0..1	The <code>Octet_Total_Count</code> property specifies the number of octets in packets in forward direction of flow. May be encoded in 4 octets using IPFIX reduced-length encoding.
Packet_Total_Count	cyboxCommon:	0..1	The <code>Packet_Total_Count</code> property specifies the

	IntegerObjectPropertyType		number of packets in forward direction of flow.
Flow_End_Reason	cyboxCommon: HexBinaryObjectPropertyType	0..1	<p>The <code>Flow_End_Reason</code> property specifies the reason for Flow termination. It may contain SiLK-specific tags. The range of values may include the following:</p> <ul style="list-style-type: none"> • 0x01: idle timeout (the Flow was terminated because it was considered to be idle). • 0x02: active timeout (the Flow was terminated for reporting purposes while it was still active, for example, after the maximum lifetime of unreported Flows was reached). • 0x03: end of Flow detected (the Flow was terminated because the Metering Process detected signals indicating the end of the Flow, for example, the TCP FIN flag.) • 0x04: forced end (the Flow was terminated because of some external event, for example, a shutdown of the Metering Process initiated by a network management application.) • 0x05: lack of resources (the Flow was terminated because of lack of resources available to the Metering Process and/or the Exporting Process.) <p>See http://www.iana.org/assignments/ipfix/ipfix.xml for more information.</p>
SiLK_App_Label	cyboxCommon: IntegerObjectPropertyType	0..1	<p>The <code>SiLK_App_Label</code> property specifies the port number that is traditionally used for that type of traffic (see the <code>/etc/services</code> file on most UNIX systems). For example, traffic that the flow generator recognizes as FTP will have a value of 21, even if that traffic is being routed through the standard HTTP/web port (80).</p>
Payload_Entropy	cyboxCommon: IntegerObjectPropertyType	0..1	<p>The <code>Payload_Entropy</code> property specifies the Shannon Entropy calculation of the forward payload data. The calculation generates a real number value between 0.0 and 8.0. That number is then converted into an 8-bit</p>

			integer value between 0 and 255. Roughly, numbers above 230 are generally compressed (or encrypted) and numbers centered around approximately 140 are English text. Lower numbers carry even less information content.
ML_App_Label	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>ML_App_Label</code> property specifies the machine-learning app label.
TCP_Flow	YAFTCPFlowType	0..1	The <code>TCP_Flow</code> property specifies the TCP-related information of the network flow.
Vlan_ID_MAC_Addr	AddressObj:AddressObjectType	0..1	The <code>Vlan_ID_MAC_Addr</code> property specifies the MAC address.
Passive_OS_Fingerprinting	cyboxCommon: PlatformSpecificationType	0..1	The <code>Passive_OS_Fingerprinting</code> property specifies the OS name and version.
First_Packet_Banner	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>First_Packet_Banner</code> property specifies the first forward packet IP payload.
Second_Packet_Banner	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>Second_Packet_Banner</code> property specifies the second forward packet IP payload.
N_Bytes_Payload	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>N_Bytes_Payload</code> property specifies the initial n bytes of forward direction of applications payload.

3.10.2 YAFReverseFlowType Class

The `YAFReverseFlowType` class specifies the properties that correspond to the reverse flow captured by a YAF record.

The property table of the YAFReverseFlowType class is given in [Table 3-44](#).

Table 3-44. Properties of the YAFReverseFlowType class

Name	Type	Multiplicity	Description
Reverse_Octet_Total_Count	cyboxCommon: IntegerObjectPropertyType	0..1	The Reverse_Octet_Total_Count property specifies the number of octets in packets in reverse direction of flow. May be encoded in 4 octets using IPFIX reduced-length encoding.
Reverse_Packet_Total_Count	cyboxCommon: IntegerObjectPropertyType	0..1	The Reverse_Packet_Total_Count property specifies the number of packets in reverse direction of flow.
Reverse_Payload_Entropy	cyboxCommon: IntegerObjectPropertyType	0..1	The Reverse_Payload_Entropy property specifies the Shannon Entropy calculation of the reverse payload data. The calculation generates a real number value between 0.0 and 8.0. That number is then converted into an 8-bit integer value between 0 and 255. Roughly, numbers above 230 are generally compressed (or encrypted) and numbers centered around approximately 140 are English text. Lower numbers carry even less information content.
Reverse_Flow_Delta_Milliseconds	cyboxCommon: IntegerObjectPropertyType	0..1	The Reverse_Flow_Delta_Milliseconds property specifies the RTT of initial handshake.
TCP_Reverse_Flow	YAFTCPFlowType	0..1	The TCP_Reverse_Flow property specifies the associated properties related to the reverse packets of the flow.
Reverse_Vlan_ID_MAC_Addr	AddressObj:AddressObjectType	0..1	The Reverse_Vlan_ID_MAC_Addr property specifies the reverse MAC address.
Reverse_Passive_OS_Fingerprinting	cyboxCommon: PlatformSpecificationType	0..1	The Reverse_Passive_OS_Fingerprinting property specifies the OS name and version of the reverse flow.

Reverse_First_Packet	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Reverse_First_Packet property specifies First reverse packet IP payload.
Reverse_N_Bytes_Payload	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Reverse_N_Bytes_Payload property specifies the initial n bytes of reverse direction of flow payload.

3.10.3 YAFTCPFlowType Class

The YAFTCPFlowType class specifies the TCP-related information of the network flow.

The property table of the YAFTCPFlowType class is given in [Table 3-45](#).

Table 3-45. Properties of the YAFTCPFlowType class

Name	Type	Multiplicity	Description
TCP_Sequence_Number	cyboxCommon: IntegerObjectPropertyType	0..1	The TCP_Sequence_Number property specifies the TCP sequence number.
Initial_TCP_Flags	PacketObj:TCPFlagsType	0..1	The Initial_TCP_Flags property specifies the TCP flags of the first packet.
Union_TCP_Flags	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Union_TCP_Flags property specifies the union of the TCP flags of the 2...nth packet.

3.11 NetflowV9FieldTypeEnum Enumeration

The literals of the NetflowV9FieldTypeEnum enumeration are given in [Table 3-46](#).

Table 3-46. Literals of the NetflowV9FieldTypeEnum enumeration

Enumeration Literal	Description
IN_BYTES(1)	The IN_BYTES(1) field represents the incoming counter with length N x 8 bits for number of bytes associated with an IP Flow.
IN_PKTS(2)	The IN_PKTS(2) field represents the incoming counter with length N x 8 bits for the number of packets associated with an IP Flow.
FLOWS(3)	The FLOWS(3) field represents the number of flows that were aggregated; default for N is 4.
PROTOCOL(4)	The PROTOCOL(4) field represents the IP protocol byte.
SRC_TOS(5)	The TOS(5) field represents the Type of Service byte setting when entering incoming interface.
TCP_FLAGS(6)	The TCP_FLAGS(6) field is cumulative of all the TCP flags seen for this flow.
L4_SRC_PORT(7)	The L4_SRC_PORT(7) field represents the TCP/UDP source port number i.e.: FTP, Telnet, or equivalent.
IPV4_SRC_ADDR(8)	The IPV4_SRC_ADDR(8) field represents the IPv4 source address.
SRC_MASK(9)	The SRC_MASK(9) field represents the number of contiguous bits in the source address subnet mask i.e.: the submask in slash notation.
INPUT_SNMP(10)	The INPUT_SNMP(10) field represents the number of contiguous bits in the source address subnet mask i.e.: the submask in slash notation.
L4_DST_PORT(11)	The LP_DST_PORT(11) field represents the TCP/UDP destination

	port number i.e.: FTP, Telnet, or equivalent.
IPv4_DST_ADDR(12)	The IPv4_DST_ADDR(12) field represents the IPv4 destination address.
DST_MASK(13)	The DST_MASK(13) field represents the number of contiguous bits in the destination address subnet mask i.e.: the submask in slash notation.
OUTPUT_SNMP(14)	The OUTPUT_SNMP(14) field represents the output interface index; default for N is 2 but higher values could be used.
IPv4_NEXT_HOP(15)	The IPv4_NEXT_HOP(15) field represents the IPv4 address of next-hop router.
SRC_AS(16)	The SRC_AS(16) field represents the source BGP autonomous system number where N could be 2 or 4.
DST_AS(17)	The DST_AS(17) field represents the destination BGP autonomous system number where N could be 2 or 4.
BGP_IPv4_NEXT_HOP(18)	The BGP_IPv4_NEXT_HOP(18) field represents the next-hop router's IP in the BGP domain.
MUL_DST_PKTS(19)	The MUL_DST_PKTS(19) field represents the IP multicast outgoing packet counter with length N x 8 bits for packets associated with the IP Flow.
MUL_DST_BYTES(20)	The MUL_DST_BYTES(20) field represents the IP multicast outgoing byte counter with length N x 8 bits for bytes associated with the IP Flow.

3.12 NetflowV9ScopeFieldTypeEnum Enumeration

The literals of the `NetflowV9ScopeFieldTypeEnum` enumeration are given in [Table 3-47](#).

Table 3-47. Literals of the `NetflowV9ScopeFieldTypeEnum` enumeration

Enumeration Literal	Description
System(1)	Indicates the System scope field type.
Interface(2)	Indicates the Interface scope field type.
LineCard(3)	Indicates the Line Card scope field type.
Cache(4)	Indicates the Netflow Cache scope field type.
Template(5)	Describes the Template scope field type.

3.13 SiLKFlowAttributesTypeEnum Enumeration

The literals of the `SiLKFlowAttributesTypeEnum` enumeration are given in [Table 3-48](#).

Table 3-48. Literals of the `SiLKFlowAttributesTypeEnum` enumeration

Enumeration Literal	Description
F (FIN flag)	Indicates that the flow generator saw additional packets in this flow following a packet with a FIN flag (excluding ACK packets).
T (Timeout)	Indicates that the flow generator prematurely created a record for a long-running connection due to a timeout. (When the flow generator <code>yaf(1)</code> is run with the <code>--silk</code> switch, it will prematurely create a flow and mark it with T

	if the byte count of the flow cannot be stored in a 32-bit value.).
C (Continuation)	Indicates that the flow generator created this flow as a continuation of long-running connection, where the previous flow for this connection met a timeout (or a byte threshold in the case of yaf).

3.14 SiLKAddressTypeEnum Enumeration

The literals of the `SiLKAddressTypeEnum` enumeration are given in [Table 3-49](#).

Table 3-49. Literals of the `SiLKAddressTypeEnum` enumeration

Enumeration Literal	Description
non-routable (0)	Denotes a (non-routable) IP address.
internal(1)	Denotes an IP address internal to the monitored network.
routable_external(2)	Denotes an IP address external to the monitored network.

3.15 SiLKDirectionTypeEnum Enumeration

The literals of the `SiLKDirectionTypeEnum` enumeration are given in [Table 3-50](#).

Table 3-50. Literals of the `SiLKDirectionTypeEnum` enumeration

Enumeration Literal	Description
in	Denotes inbound traffic relative to a sensor.

inweb	Denotes inbound web traffic relative to a sensor. SiLK categorizes a flow as web if the protocol is TCP and either the source port or destination port is one of 80, 443, or 8080.
innull	Denotes null inbound traffic relative to a sensor.
out	Denotes outbound traffic relative to a sensor.
outweb	Denotes outbound web traffic relative to a sensor. SiLK categorizes a flow as web if the protocol is TCP and either the source port or destination port is one of 80, 443, or 8080.
outnull	Denotes null outbound traffic relative to a sensor.

3.16 SiLKSensorClassTypeEnum Enumeration

The literals of the `SiLKSensorClassTypeEnum` enumeration are given in [Table 3-51](#).

Table 3-51. Literals of the `SiLKSensorClassTypeEnum` enumeration

Enumeration Literal	Description
all	Defines sensor class "all".

4 Conformance

Implementations have discretion over which parts (components, properties, extensions, controlled vocabularies, etc.) of CybOX they implement (e.g., Observable/Object).

[1] Conformant implementations must conform to all normative structural specifications of the UML model or additional normative statements within this document that apply to the portions of CybOX they implement (e.g., implementers of the entire Observable class must conform to all normative structural specifications of the UML model regarding the Observable class or additional normative statements contained in the document that describes the Observable class).

[2] Conformant implementations are free to ignore normative structural specifications of the UML model or additional normative statements within this document that do not apply to the portions of CybOX they implement (e.g., non-implementers of any particular properties of the Observable class are free to ignore all normative structural specifications of the UML model regarding those properties of the Observable class or additional normative statements contained in the document that describes the Observable class).

The conformance section of this document is intentionally broad and attempts to reiterate what already exists in this document.

Appendix A. Acknowledgments

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

Aetna

David Crawford

AIT Austrian Institute of Technology

Roman Fiedler

Florian Skopik

Australia and New Zealand Banking Group (ANZ Bank)

Dean Thompson

Blue Coat Systems, Inc.

Owen Johnson

Bret Jordan

Century Link

Cory Kennedy

CIRCL

Alexandre Dulaunoy

Andras Iklody

Raphaël Vinot

Citrix Systems

Joey Peloquin

Dell

Will Urbanski

Jeff Williams

DTCC

Dan Brown

Gordon Hundley

Chris Koutras

EMC

Robert Griffin

Jeff Odom

Ravi Sharda

Financial Services Information Sharing and Analysis Center (FS-ISAC)

David Eilken

Chris Ricard

Fortinet Inc.

Gavin Chow

Airbus Group SAS

Joerg Eschweiler

Marcos Orallo

Anomali

Ryan Clough

Wei Huang

Hugh Njemanze

Katie Pelusi

Aaron Shelmire

Jason Trost

Bank of America

Alexander Foley

Center for Internet Security (CIS)

Sarah Kelley

Check Point Software Technologies

Ron Davidson

Cisco Systems

Syam Appala

Ted Bedwell

David McGrew

Pavan Reddy

Omar Santos

Jyoti Verma

Cyber Threat Intelligence Network, Inc. (CTIN)

Doug DePeppe

Jane Ginn

Ben Othman

DHS Office of Cybersecurity and Communications (CS&C)

Richard Struse

Marlon Taylor

EclecticIQ

Marko Dragoljevic

Joep Gommers

Sergey Polzunov

Kenichi Terashita
Fujitsu Limited
Neil Edwards
Frederick Hirsch
Ryusuke Masuoka
Daisuke Murabayashi

Google Inc.
Mark Risher

Hitachi, Ltd.
Kazuo Noguchi
Akihito Sawada
Masato Terada

iboss, Inc.
Paul Martini

Individual
Jerome Athias
Peter Brown
Elysa Jones
Sanjiv Kalkar
Bar Lockwood
Terry MacDonald
Alex Pinto

Intel Corporation
Tim Casey
Kent Landfield

JPMorgan Chase Bank, N.A.
Terrence Driscoll
David Laurance

LookingGlass
Allan Thomson
Lee Vorthman

Mitre Corporation
Greg Back
Jonathan Baker
Sean Barnum
Desiree Beck
Nicole Gong
Jasen Jacobsen
Ivan Kirillov
Richard Piazza
Jon Salwen

Rutger Prins
Andrei Sirghi
Raymon van der Velde

eSentire, Inc.
Jacob Gajek

FireEye, Inc.
Phillip Boles
Pavan Gorakav
Anuj Kumar
Shyamal Pandya
Paul Patrick
Scott Shreve

Fox-IT
Sarah Brown

Georgetown University
Eric Burger

Hewlett Packard Enterprise (HPE)
Tomas Sander

IBM
Peter Allor
Eldan Ben-Haim
Sandra Hernandez
Jason Keirstead
John Morris
Laura Rusu
Ron Williams

IID
Chris Richardson

Integrated Networking Technologies, Inc.
Patrick Maroney

Johns Hopkins University Applied Physics Laboratory
Karin Marr
Julie Modlin
Mark Moss
Pamela Smith

Kaiser Permanente
Russell Culpepper
Beth Pumo

Lumeta Corporation
Brandon Hoffman

MTG Management Consultants, LLC.

Charles Schmidt
Emmanuelle Vargas-Gonzalez
John Wunder
National Council of ISACs (NCI)
Scott Algeier
Denise Anderson
Josh Poster
NEC Corporation
Takahiro Kakumaru
North American Energy Standards Board
David Darnell
Object Management Group
Cory Casanave
Palo Alto Networks
Vishaal Hariprasad
Queralt, Inc.
John Tolbert
Resilient Systems, Inc.
Ted Julian
Securonix
Igor Baikalov
Siemens AG
Bernd Grobauer
Soltra
John Anderson
Aishwarya Asok Kumar
Peter Ayasse
Jeff Beekman
Michael Butt
Cynthia Camacho
Aharon Chernin
Mark Clancy
Brady Cotton
Trey Darley
Mark Davidson
Paul Dion
Daniel Dye
Robert Hutto
Raymond Keckler
Ali Khan
Chris Kiehl

James Cabral
National Security Agency
Mike Boyle
Jessica Fitzgerald-McKay
New Context Services, Inc.
John-Mark Gurney
Christian Hunt
James Moler
Daniel Riedel
Andrew Storms
OASIS
James Bryce Clark
Robin Cover
Chet Ensign
Open Identity Exchange
Don Thibeau
PhishMe Inc.
Josh Larkins
Raytheon Company-SAS
Daniel Wyschogrod
Retail Cyber Intelligence Sharing Center (R-CISC)
Brian Engle
Semper Fortis Solutions
Joseph Brand
Splunk Inc.
Cedric LeRoux
Brian Luger
Kathy Wang
TELUS
Greg Reaume
Alan Steer
Threat Intelligence Pty Ltd
Tyron Miller
Andrew van der Stock
ThreatConnect, Inc.
Wade Baker
Cole Iliff
Andrew Pendergast
Ben Schmoker
Jason Spies
TruSTAR Technology

Clayton Long
Michael Pepin
Natalie Suarez
David Waters
Benjamin Yates

Symantec Corp.

Curtis Kostrosky

The Boeing Company

Crystal Hayes

ThreatQuotient, Inc.

Ryan Trost

U.S. Bank

Mark Angel
Brad Butts
Brian Fay
Mona Magathan
Yevgen Sautin

US Department of Defense (DoD)

James Bohling
Eoghan Casey
Gary Katz
Jeffrey Mates

VeriSign

Robert Coderre
Kyle Maxwell
Eric Osterweil

Chris Roblee

United Kingdom Cabinet Office

Iain Brown
Adam Cooper
Mike McLellan
Chris O'Brien
James Penman
Howard Staple
Chris Taylor
Laurie Thomson
Alastair Treharne
Julian White
Bethany Yates

US Department of Homeland Security

Evette Maynard-Noel
Justin Stekervetz

ViaSat, Inc.

Lee Chieffalo
Wilson Figueroa
Andrew May

Yaana Technologies, LLC

Anthony Rutkowski

The authors would also like to thank the larger CybOX Community for its input and help in reviewing this document.

Appendix B. Revision History

Revision	Date	Editor	Changes Made
wd01	15 December 2015	Desiree Beck Trey Darley Ivan Kirillov Rich Piazza	Initial transfer to OASIS template