CybOXTM Version 2.1.1 Part 37: Network Flow Object

Working Draft 01

15 December 2015

Technical Committee:

[OASIS Cyber Threat Intelligence (CTI) TC](https://www.oasis-open.org/committees/cti)

Chair:

Richard Struse ([Richard.Struse@HQ.DHS.GOV](mailto:Richard.Struse@HQ.DHS.GOV)), [DHS Office of Cybersecurity and Communications (CS&C)](http://www.dhs.gov/office-cybersecurity-and-communications)

Editors:

Desiree Beck ([dbeck@mitre.org](mailto:ikirillov@mitre.org)), [MITRE Corporation](http://www.mitre.org/)

Trey Darley ([trey@soltra.com](mailto:trey@soltra.com)), [Soltra](http://www.soltra.com/)

Ivan Kirillov ([ikirillov@mitre.org](mailto:ikirillov@mitre.org)), [MITRE Corporation](http://www.mitre.org/)

Rich Piazza ([rpiazza@mitre.org](mailto:ikirillov@mitre.org)), [MITRE Corporation](http://www.mitre.org/)

Additional artifacts:

This prose specification is one component of a Work Product which consists of:

* *CybOX™ Version 2.1.1 Part 01: Overview*. [URI]
* *CybOX™ Version 2.1.1 Part 02: Common*. [URI]
* *CybOX™ Version 2.1.1 Part 03: Core*. [URI]
* *CybOX™ Version 2.1.1 Part 04: Default Extensions*. [URI]
* *CybOX™ Version 2.1.1 Part 05: Default Vocabularies*. [URI]
* *CybOX™ Version 2.1.1 Part 06: UML Model*. [URI]
* *CybOX™ Version 2.1.1 Part 07: API Object*. [URI]
* *CybOX™ Version 2.1.1 Part 08: ARP Cache Object*. [URI]
* *CybOX™ Version 2.1.1 Part 09: AS Object*. [URI]
* *CybOXTM Version 2.1.1 Part 10: Account Object*. [URI]
* *CybOXTM Version 2.1.1 Part 11: Address Object*. [URI]
* *CybOXTM Version 2.1.1 Part 12: Archive File Object*. [URI]
* *CybOXTM Version 2.1.1 Part 13: Artifact Object*. [URI]
* *CybOXTM Version 2.1.1 Part 14: Code Object*. [URI]
* *CybOXTM Version 2.1.1 Part 15: Custom Object*. [URI]
* *CybOXTM Version 2.1.1 Part 16: DNS Cache Object*. [URI]
* *CybOXTM Version 2.1.1 Part 17: DNS Query Object*. [URI]
* *CybOXTM Version 2.1.1 Part 18: DNS Record Object*. [URI]
* *CybOXTM Version 2.1.1 Part 19: Device Object*. [URI]
* *CybOXTM Version 2.1.1 Part 20: Disk Object*. [URI]
* *CybOXTM Version 2.1.1 Part 21: Disk Partition Object*. [URI]
* *CybOXTM Version 2.1.1 Part 22: Domain Name Object*. [URI]
* *CybOXTM Version 2.1.1 Part 23: Email Message Object*. [URI]
* *CybOXTM Version 2.1.1 Part 24: File Object*. [URI]
* *CybOXTM Version 2.1.1 Part 25: GUI Dialogbox Object*. [URI]
* *CybOXTM Version 2.1.1 Part 26: GUI Object*. [URI]
* *CybOXTM Version 2.1.1 Part 27: GUI Window Object*. [URI]
* *CybOXTM Version 2.1.1 Part 28: HTTP Session Object*. [URI]
* *CybOXTM Version 2.1.1 Part 29: Hostname Object*. [URI]
* *CybOXTM Version 2.1.1 Part 30: Image File Object*. [URI]
* *CybOXTM Version 2.1.1 Part 31: Library File Object*. [URI]
* *CybOXTM Version 2.1.1 Part 32: Link Object*. [URI]
* *CybOXTM Version 2.1.1 Part 33: Linux Package Object*. [URI]
* *CybOXTM Version 2.1.1 Part 34: Memory Object*. [URI]
* *CybOXTM Version 2.1.1 Part 35: Mutex Object*. [URI]
* *CybOXTM Version 2.1.1 Part 36: Network Connection Object*. [URI]
* *CybOXTM Version 2.1.1 Part 37: Network Flow Object*. (this document)
* *CybOXTM Version 2.1.1 Part 38: Network Packet Object*. [URI]
* *CybOXTM Version 2.1.1 Part 39: Network Route Entry Object*. [URI]
* *CybOXTM Version 2.1.1 Part 40: Network Route Object*. [URI]
* *CybOXTM Version 2.1.1 Part 41: Network Socket Object*. [URI]
* *CybOXTM Version 2.1.1 Part 42: Network Subnet Object*. [URI]
* *CybOXTM Version 2.1.1 Part 43: PDF File Object*. [URI]
* *CybOXTM Version 2.1.1 Part 44: Pipe Object*. [URI]
* *CybOXTM Version 2.1.1 Part 45: Port Object*. [URI]
* *CybOXTM Version 2.1.1 Part 46: Process Object*. [URI]
* *CybOXTM Version 2.1.1 Part 47: Product Object*. [URI]
* *CybOXTM Version 2.1.1 Part 48: SMS Message Object*. [URI]
* *CybOXTM Version 2.1.1 Part 49: Semaphore Object*. [URI]
* *CybOXTM Version 2.1.1 Part 50: Socket Address Object*. [URI]
* *CybOXTM Version 2.1.1 Part 51: System Object*. [URI]
* *CybOXTM Version 2.1.1 Part 52: URI Object*. [URI]
* *CybOXTM Version 2.1.1 Part 53: URL History Object*. [URI]
* *CybOXTM Version 2.1.1 Part 54: Unix File Object*. [URI]
* *CybOXTM Version 2.1.1 Part 55: Unix Network Route Entry Object*. [URI]
* *CybOXTM Version 2.1.1 Part 56: Unix Pipe Object*. [URI]
* *CybOXTM Version 2.1.1 Part 57: Unix Process Object*. [URI]
* *CybOXTM Version 2.1.1 Part 58: Unix User Account Object*. [URI]
* *CybOXTM Version 2.1.1 Part 59: Unix Volume Object*. [URI]
* *CybOXTM Version 2.1.1 Part 60: User Account Object*. [URI]
* *CybOXTM Version 2.1.1 Part 61: User Session Object*. [URI]
* *CybOXTM Version 2.1.1 Part 62: Volume Object*. [URI]
* *CybOXTM Version 2.1.1 Part 63: Whois Object*. [URI]
* *CybOXTM Version 2.1.1 Part 64: Win Computer Account Object*. [URI]
* *CybOXTM Version 2.1.1 Part 65: Win Critical Section Object*. [URI]
* *CybOXTM Version 2.1.1 Part 66: Win Driver Object*. [URI]
* *CybOXTM Version 2.1.1 Part 67: Win Event Log Object*. [URI]
* *CybOXTM Version 2.1.1 Part 68: Win Event Object*. [URI]
* *CybOXTM Version 2.1.1 Part 69: Win Executable File Object*. [URI]
* *CybOXTM Version 2.1.1 Part 70: Win File Object*. [URI]
* *CybOXTM Version 2.1.1 Part 71: Win Filemapping Object*. [URI]
* *CybOXTM Version 2.1.1 Part 72: Win Handle Object*. [URI]
* *CybOXTM Version 2.1.1 Part 73: Win Hook Object*. [URI]
* *CybOXTM Version 2.1.1 Part 74: Win Kernel Hook Object*. [URI]
* *CybOXTM Version 2.1.1 Part 75: Win Kernel Object*. [URI]
* *CybOXTM Version 2.1.1 Part 76: Win Mailslot Object*. [URI]
* *CybOXTM Version 2.1.1 Part 77: Win Memory Page Region Object*. [URI]
* *CybOXTM Version 2.1.1 Part 78: Win Mutex Object*. [URI]
* *CybOXTM Version 2.1.1 Part 79: Win Network Route Entry Object*. [URI]
* *CybOXTM Version 2.1.1 Part 80: Win Network Share Object*. [URI]
* *CybOXTM Version 2.1.1 Part 81: Win Pipe Object*. [URI]
* *CybOXTM Version 2.1.1 Part 82: Win Prefetch Object*. [URI]
* *CybOXTM Version 2.1.1 Part 83: Win Process Object*. [URI]
* *CybOXTM Version 2.1.1 Part 84: Win Registry Key Object*. [URI]
* *CybOXTM Version 2.1.1 Part 85: Win Semaphore Object*. [URI]
* *CybOXTM Version 2.1.1 Part 86: Win Service Object*. [URI]
* *CybOXTM Version 2.1.1 Part 87: Win System Object*. [URI]
* *CybOXTM Version 2.1.1 Part 88: Win System Restore Object*. [URI]
* *CybOXTM Version 2.1.1 Part 89: Win Task Object*. [URI]
* *CybOXTM Version 2.1.1 Part 90: Win Thread Object*. [URI]
* *CybOXTM Version 2.1.1 Part 91: Win User Account Object*. [URI]
* *CybOXTM Version 2.1.1 Part 92: Win Volume Object*. [URI]
* *CybOXTM Version 2.1.1 Part 93: Win Waitable Timer Object*. [URI]
* *CybOXTM Version 2.1.1 Part 94: X509 Certificate Object*. [URI]

Related work:

This specification is related to:

* *STIXTM Version 1.2.1 (placeholder)*

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 [Working Draft](https://www.oasis-open.org/policies-guidelines/tc-process#dWorkingDraft) (WD) has been produced by one or more TC Members; it has not yet been voted on by the TC or [approved](https://www.oasis-open.org/policies-guidelines/tc-process#committeeDraft) as a Committee Draft (Committee Specification Draft or a Committee Note Draft). The OASIS document [Approval Process](https://www.oasis-open.org/policies-guidelines/tc-process#standApprovProcess) begins officially with a TC vote to approve a WD as a Committee Draft. A TC may approve a Working Draft, revise it, and re-approve it any number of times as a Committee Draft.

URI patterns:

Initial publication URI:  
http://docs.oasis-open.org/cti/stix/v1.2.1/csd01/part1-overview/stix-v1.2.1-csd01-part1-overview.docx

Permanent “Latest version” URI:  
http://docs.oasis-open.org/cti/stix/v1.2.1/stix-v1.2.1-part1-overview.docx

(Managed by OASIS TC Administration; please don’t modify.)

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](https://www.oasis-open.org/policies-guidelines/ipr) 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.

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](#_Toc449967465)

[1.1 CybOXTM Specification Documents 7](#_Toc449967466)

[1.2 Document Conventions 7](#_Toc449967467)

[1.2.1 Fonts 7](#_Toc449967468)

[1.2.2 UML Package References 8](#_Toc449967469)

[1.2.3 UML Diagrams 8](#_Toc449967470)

[1.2.3.1 Class Properties 8](#_Toc449967471)

[1.2.3.2 Diagram Icons and Arrow Types 8](#_Toc449967472)

[1.2.4 Property Table Notation 9](#_Toc449967473)

[1.2.5 Property and Class Descriptions 9](#_Toc449967474)

[1.3 Terminology 10](#_Toc449967475)

[1.4 Normative References 10](#_Toc449967476)

[2 Background Information 11](#_Toc449967477)

[2.1 Cyber Observables 11](#_Toc449967478)

[2.2 Objects 11](#_Toc449967479)

[3 Data Model 12](#_Toc449967480)

[3.1 NetworkFlowObjectType Class 12](#_Toc449967481)

[3.2 NetworkLayerInfoType Class 13](#_Toc449967482)

[3.3 NetworkFlowLabelType Class 14](#_Toc449967483)

[3.4 UnidirectionalRecordType Class 15](#_Toc449967484)

[3.5 BidirectionalRecordType Class 16](#_Toc449967485)

[3.6 IPFIXMessageType Class 17](#_Toc449967486)

[3.6.1 IPFIXMessageHeaderType Class 18](#_Toc449967487)

[3.6.2 IPFIXSetType Class 19](#_Toc449967488)

[3.6.3 IPFIXTemplateSetType Class 20](#_Toc449967489)

[3.6.4 IPFIXOptionsTemplateSetType Class 21](#_Toc449967490)

[3.6.5 IPFIXDataSetType Class 22](#_Toc449967491)

[3.6.6 IPFIXSetHeaderType Class 22](#_Toc449967492)

[3.6.7 IPFIXTemplateRecordType Class 23](#_Toc449967493)

[3.6.8 IPFIXTemplateRecordHeaderType Class 24](#_Toc449967494)

[3.6.9 IPFIXTemplateRecordFieldSpecifiersType Class 24](#_Toc449967495)

[3.6.10 IPFIXOptionsTemplateRecordType Class 25](#_Toc449967496)

[3.6.11 IPFIXOptionsTemplateRecordHeaderType Class 26](#_Toc449967497)

[3.6.12 IPFIXOptionsTemplateRecordFieldSpecifiersType Class 26](#_Toc449967498)

[3.6.13 IPFIXDataRecordType Class 28](#_Toc449967499)

[3.7 NetflowV9ExportPacketType Class 29](#_Toc449967500)

[3.7.1 NetflowV9PacketHeaderType Class 30](#_Toc449967501)

[3.7.2 NetflowV9FlowSetType Class 31](#_Toc449967502)

[3.7.3 NetflowV9TemplateFlowSetType Class 32](#_Toc449967503)

[3.7.4 NetflowV9TemplateRecordType Class 33](#_Toc449967504)

[3.7.5 NetflowV9FieldType Data Type 34](#_Toc449967505)

[3.7.6 NetflowV9OptionsTemplateFlowSetType Class 34](#_Toc449967506)

[3.7.7 NetflowV9OptionsTemplateRecordType Class 34](#_Toc449967507)

[3.7.8 NetflowV9ScopeFieldType Data Type 36](#_Toc449967508)

[3.7.9 NetflowV9DataFlowSetType Class 36](#_Toc449967509)

[3.7.10 NetflowV9DataRecordType Class 37](#_Toc449967510)

[3.7.11 FlowDataRecordType Class 38](#_Toc449967511)

[3.7.12 FlowCollectionElementType Class 38](#_Toc449967512)

[3.7.13 OptionsDataRecordType Class 38](#_Toc449967513)

[3.7.14 OptionCollectionElementType Class 39](#_Toc449967514)

[3.8 NetflowV5PacketType Class 40](#_Toc449967515)

[3.8.1 NetflowV5FlowHeaderType Class 41](#_Toc449967516)

[3.8.2 NetflowV5FlowRecordType Class 42](#_Toc449967517)

[3.9 SiLKRecordType Class 44](#_Toc449967518)

[3.9.1 SiLKFlowAttributesType Data Type 47](#_Toc449967519)

[3.9.2 SiLKAddressType Data Type 47](#_Toc449967520)

[3.9.3 SiLKCountryCodeType Class 47](#_Toc449967521)

[3.9.4 SiLKSensorInfoType Class 47](#_Toc449967522)

[3.9.5 SiLKDirectionType Class 48](#_Toc449967523)

[3.9.6 SiLKSensorClassType Class 48](#_Toc449967524)

[3.10 YAFRecordType Class 48](#_Toc449967525)

[3.10.1 YAFFlowType Class 49](#_Toc449967526)

[3.10.2 YAFReverseFlowType Class 51](#_Toc449967527)

[3.10.3 YAFTCPFlowType Class 52](#_Toc449967528)

[3.11 NetflowV9FieldTypeEnum Enumeration 53](#_Toc449967529)

[3.12 NetflowV9ScopeFieldTypeEnum Enumeration 55](#_Toc449967530)

[3.13 SiLKFlowAttributesTypeEnum Enumeration 55](#_Toc449967531)

[3.14 SiLKAddressTypeEnum Enumeration 56](#_Toc449967532)

[3.15 SiLKDirectionTypeEnum Enumeration 56](#_Toc449967533)

[3.16 SiLKSensorClassTypeEnum Enumeration 57](#_Toc449967534)

[4 Conformance 58](#_Toc449967535)

[Appendix A. Acknowledgments 59](#_Toc449967536)

[Appendix B. Revision History 60](#_Toc449967537)

# Introduction

[All text is normative unless otherwise labeled]

The Cyber Observable Expression (CybOXTM) 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**.

## CybOXTM 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*](#AdditionalArtifacts) 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*](#AdditionalArtifacts) also summarizes the relationship of CybOX to other languages, and outlines general CybOX data model conventions.

## Document Conventions

The following conventions are used in this document.

### 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*](#AdditionalArtifacts).

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 (withsingle 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*

### 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*](#AdditionalArtifacts) 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.

### 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.

#### 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.

#### 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. |

### 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.

### Property and Class Descriptions

Each class and property defined in CybOX is described using the format, “The X property verbY.” 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** |
| captures | 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. |
|  | *Examples*:  The Observable\_Source property characterizes the source of the Observable information. Examples of details captured include identifying characteristics, time-related attributes, and a list of the tools used to collect the information.  The Description property captures a textual description of the Action. |
| characterizes | 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. |
|  | *Examples*:  The Action property characterizes a cyber observable Action.  The Obfuscation\_Technique property characterizes a technique an attacker could potentially leverage to obfuscate the Observable. |
| specifies | 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. |
|  | *Example*:  The cybox\_major\_version property specifies the major version of the CybOX language used for the set of Observables. |

## 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**]**.

## 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>.

# 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 **3Error! Reference source not found.**.

## 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.

## 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.

# Data Model

## 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).

In CybOX 2.1.1 the properties Unidirectional\_Flow\_Record and Bidirectional\_Flow\_Record are mutually exclusive, i.e., only one property can be populated. This restriction is based on the fact that the classes UnidirectionalRecordType, and BidirectionalRecordType can be modelled as subclasses of the NetworkFlowObjectType class. In future releases, it will probably be modelled that way.

The UML diagram corresponding to the NetworkFlowObjectType class is shown in **Figure 3‑1**.

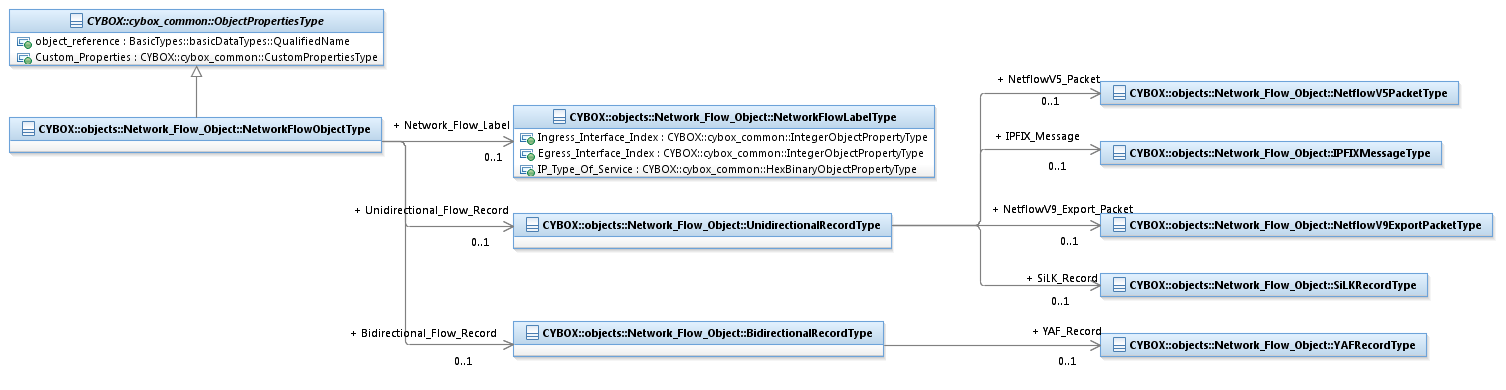


Figure 3‑1. UML diagram of the NetworkFlowObjectType class

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 Network\_Flow\_Label 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. |
| **Unidirectional\_Flow\_Record** | UnidirectionalRecordType | 0..1 | The Unidirectional\_Flow\_Record property represents flow-record formats that capture data in one direction only (e.g., Netflow v9).  Only one of the Unidirectional\_Flow\_Record and Bidirectional\_Flow\_Record properties can be populated. |
| **Bidirectional\_Flow\_Record** | BidirectionalRecordType | 0..1 | The Bidirectional\_Flow\_Record property represents flow-record formats that capture data in both directions (e.g., YAF).  Only one of the Unidirectional\_Flow\_Record and Bidirectional\_Flow\_Record properties can be populated. |

## 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 3‑2**.

Table 3‑2. Properties of the NetworkLayerInfoType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Src\_Socket\_Address** | SocketAddressObj:  SocketAddressObjectType | 0..1 | The Src\_Socket\_Address 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 Dest\_Socket\_Address 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:  IANAAssignedIPNumbersType | 0..1 | The IP\_Protocol property specifies the IP Protocol of the network flow. This is usually TCP, UDP, or SCTP, but can include 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](http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xml%20) for reference. |

## 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 3‑3**.

Table 3‑3. Properties of the NetworkFlowLabelType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Ingress\_Interface\_Index** | cyboxCommon:  IntegerObjectPropertyType | 0..1 | The Ingress\_Interface\_Index 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 Egress\_Interface\_Index 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 IP\_Type\_Of\_Service property specifies the type of service (ToS) property from the IP header. See <http://tools.ietf.org/html/rfc1349.txt> for more information. |

## UnidirectionalRecordType Class

The UnidirectionalRecordType class specifies the netflow record formats that capture traffic in one direction.

In CybOX 2.1.1, all of the properties of the UnidirectionalRecordType class are mutually exclusive, i.e., only one property can be populated. This restriction is based on the fact that the classes IPFIXMessageType, NetflowV9ExportPacketType, NetflowV5PacketType and SiLKRecordType can be modelled as subclasses of the UnidirectionalRecordType class. In future releases, it will probably be modelled that way.

The property table of the UnidirectionalRecordType class is given in **Table 3‑4**.

Table 3‑4. Properties of the UnidirectionalRecordType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **IPFIX\_Message** | IPFIXMessageType | 0..1 | The IPFIX\_Message 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.  Only one of the UnidirectionalRecordType properties can be populated. |
| **NetflowV9\_Export\_Packet** | NetflowV9ExportPacketType | 0..1 | The NetflowV9\_Export\_Packet property represents the Netflow V9 flow record format. See <https://www.ietf.org/rfc/rfc3954.txt> (Netflow v9) for more information.  Only one of the UnidirectionalRecordType properties can be populated. |
| **NetflowV5\_Packet** | NetflowV5PacketType | 0..1 | The NetflowV5\_Packet property represents the Netflow v5 flow record format, which is commonly used to represent network flow data.  Only one of the UnidirectionalRecordType properties can be populated. |
| **SiLK\_Record** | SiLKRecordType | 0..1 | The SiLK\_Record 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](http://tools.netsa.cert.org/silk/analysis-handbook.pdf%20) for more information.  Only one of the UnidirectionalRecordType properties can be populated. |

## 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 3‑5**.

Table 3‑5. 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](http://www.usenix.org/event/lisa10/tech/full_papers/Inacio.pdf%20) or [http://tools.netsa.cert.org/yaf/index.html](http://tools.netsa.cert.org/yaf/index.html%20) for more information. |

## IPFIXMessageType Class

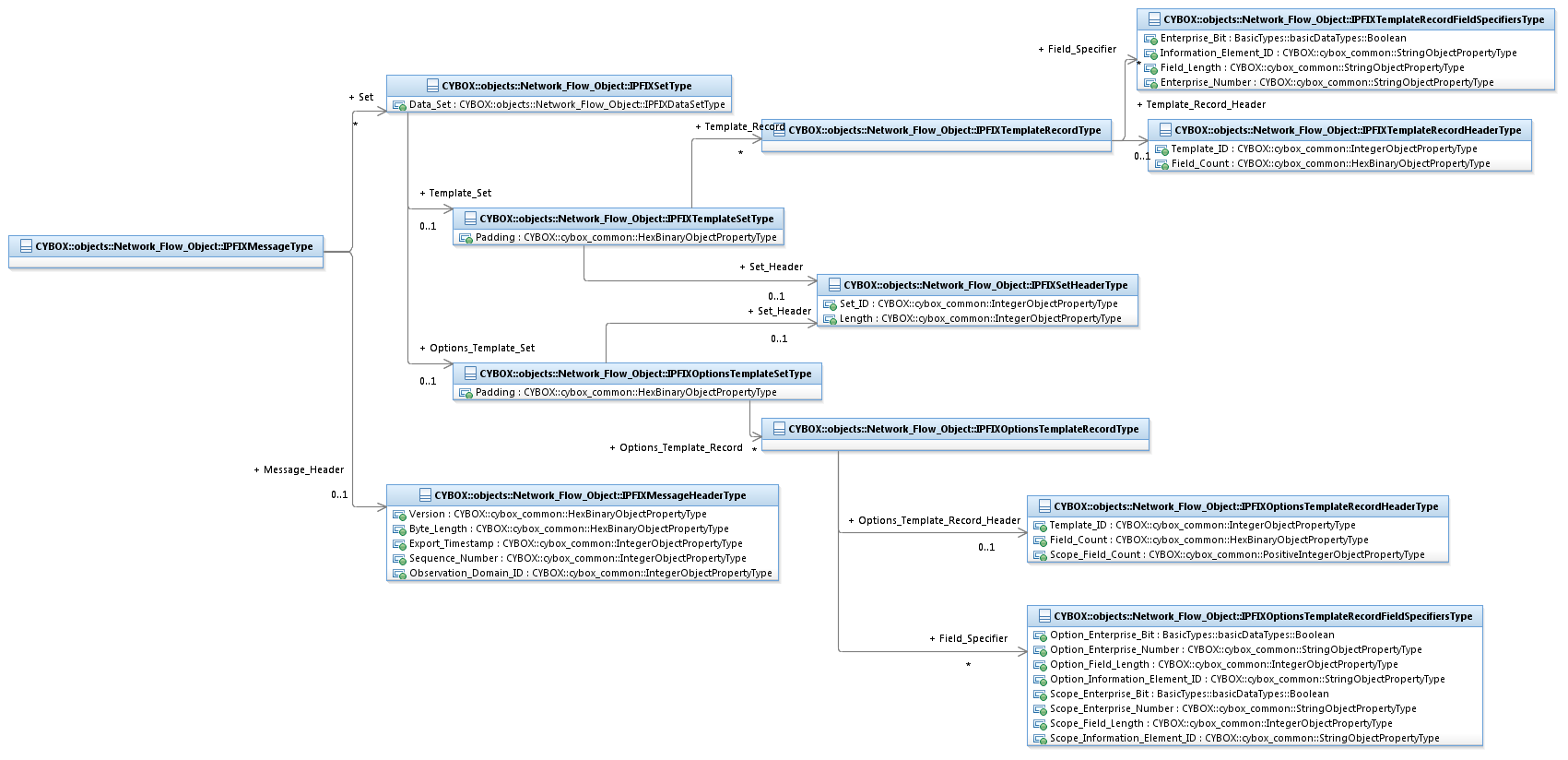


Figure 3‑2. UML diagram of the IPFIXMessageType class

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 3‑6.

Table 3‑6. Properties of the IPFIXMessageType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Message\_Header** | IPFIXMessageHeaderType | 0..1 | The Message\_Header 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 Set property Set 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. |

### 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 3‑7**.

Table 3‑7. Properties of the IPFIXMessageHeaderType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Version** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Version 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 Byte\_Length 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 Export\_Timestamp 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 Sequence\_Number 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 Observation\_Domain\_ID 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. |

### 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.

In CybOX 2.1.1, all of the properties of the IPFIXSetType class are mutually exclusive, i.e., only one property can be populated. This restriction is based on the fact that the classes IPFIXTemplateSetType, IPFIXOptionsTemplateSetType and IPFIXDataSetType can be modelled as subclasses of the IPFIXSetType class. In future releases, it will probably be modelled that way.

The property table of the IPFIXSetType class is given in **Table 3‑8**.

Table 3‑8. Properties of the IPFIXSetType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Template\_Set** | IPFIXTemplateSetType | 0..1 | The Template\_Set property indicates a collection of one or more Template Records that have been grouped together in an IPFIX message.  Only one of the IPFIXSetType properties can be populated. |
| **Options\_Template\_Set** | IPFIXOptionsTemplateSetType | 0..1 | The Options\_Template\_Set property indicates a collection of one or more Options Template Records that have been grouped together in an IPFIX message.  Only one of the IPFIXSetType properties can be populated. |
| **Data\_Set** | IPFIXDataSetType | 0..1 | The Data\_Set 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.  Only one of the IPFIXSetType properties can be populated. |

### 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 3‑9**.

Table 3‑9. Properties of the IPFIXTemplateSetType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Set\_Header** | IPFIXSetHeaderType | 0..1 | The Set\_Header property indicates the Set Header region, which is 32-bit region containing the 16-bit properties Set ID and Length. |
| **Template\_Record** | IPFIXTemplateRecordType | 0..\* | The Template\_Record property indicates the region of Template Records. These are the same properties referenced in the IPFIXTemplateRecordType. |
| **Padding** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Padding 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. |

### 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 3‑10**.

Table 3‑10. Properties of the IPFIXOptionsTemplateSetType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Set\_Header** | IPFIXSetHeaderType | 0..1 | The Set\_Header 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 IPFIXSetHeaderType. |
| **Options\_Template\_Record** | IPFIXOptionsTemplateRecordType | 0..\* | The Options\_Template\_Record property indicates the region of Options Template Records. These are the same properties referenced in the IPFIXOptionsTemplateRecordType. |
| **Padding** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Padding 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. |

### 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 3‑11**.

Table 3‑11. Properties of the IPFIXDataSetType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Set\_Header** | IPFIXSetHeaderType | 0..1 | The Set\_Header 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 IPFIXSetHeaderType. |
| **Data\_Record** | IPFIXDataRecordType | 0..\* | The Data\_Record property indicates the region of Data Records, which consist of a series of property values without a header. |
| **Padding** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Padding 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. |

### IPFIXSetHeaderType Class

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

The property table of the IPFIXSetHeaderType class is given in **Table 3‑12**.

Table 3‑12. Properties of the IPFIXSetHeaderType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Set\_ID** | cyboxCommon:  IntegerObjectPropertyType | 0..1 | The Set\_ID 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** | cyboxCommon:  IntegerObjectPropertyType | 0..1 | The Length 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. |

### 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 3‑13**.

Table 3‑13. Properties of the IPFIXTemplateRecordType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Template\_Record\_**  **Header** | IPFIXTemplateRecordHeaderType | 0..1 | The Template\_Record\_Header 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 IPFIXTemplateRecordHeaderType. |
| **Field\_Specifier** | IPFIXTemplateRecordFieldSpecifiersType | 0..\* | The Field\_Specifier property indicates the region of Field Specifiers. These are the same properties referenced in the IPFIXTemplateRecordFieldSpecifiersType. |

### 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 3‑14**.

Table 3‑14. Properties of the IPFIXTemplateRecordHeaderType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Template\_ID** | cyboxCommon:  IntegerObjectPropertyType | 0..1 | The Template\_ID 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 Field\_Count property specifies the number of properties in this Template Record. |

### 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 3‑15**.

Table 3‑15. Properties of the IPFIXTemplateRecordFieldSpecifiersType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Enterprise\_Bit** | basicDataTypes:Boolean | 0..1 | The Enterprise\_Bit 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 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. |
| **Information\_Element\_ID** | cyboxCommon:  StringObjectPropertyType | 0..1 | The Information\_Element\_ID 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 Field\_Length 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 Enterprise\_Number 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. |

### 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 3‑16**.

Table 3‑16. Properties of the IPFIXOptionsTemplateRecordType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Options\_Template\_**  **Record\_Header** | IPFIXOptionsTemplateRecordHeaderType | 0..1 | The Options\_Template\_Record\_Header property indicates the Options Template Record 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 Field\_Specifier property indicates the region of Field Specifiers. These are the same properties referenced in the IPFIXOptionsTemplateRecordFieldSpecifiersType. |

### IPFIXOptionsTemplateRecordHeaderType Class

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

The property table of the IPFIXOptionsTemplateRecordHeaderType class is given in **Table 3‑17**.

Table 3‑17. Properties of the IPFIXOptionsTemplateRecordHeaderType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Template\_ID** | cyboxCommon:  IntegerObjectPropertyType | 0..1 | The Template\_ID 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 Field\_Count property specifies the number of properties in this Options Template Record, INCLUDING the Scope Fields. |
| **Scope\_Field\_Count** | cyboxCommon:  PositiveIntegerObjectPropertyType | 0..1 | The Scope\_Field\_Count 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. |

### 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 3‑18**.

Table 3‑18. Properties of the IPFIXOptionsTemplateRecordFieldSpecifiersType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Scope\_Enterprise\_Bit** | basicDataTypes:Boolean | 0..1 | The Scope\_Enterprise\_Bit 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 filed 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** | cyboxCommon:  StringObjectPropertyType | 0..1 | The Scope\_Information\_Element\_ID 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** | cyboxCommon:  IntegerObjectPropertyType | 0..1 | The Scope\_Field\_Length 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 Scope\_Enterprise\_Number 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 Option\_Enterprise\_Bit 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 Option\_Information\_Element\_ID 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 Option\_Field\_Length 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 Option\_Enterprise\_Number 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. |

### 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 3‑19**.

Table 3‑19. Properties of the IPFIXDataRecordType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Field\_Value** | cyboxCommon:  StringObjectPropertyType | 0..\* | The Field\_Value 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". |

## NetflowV9ExportPacketType Class

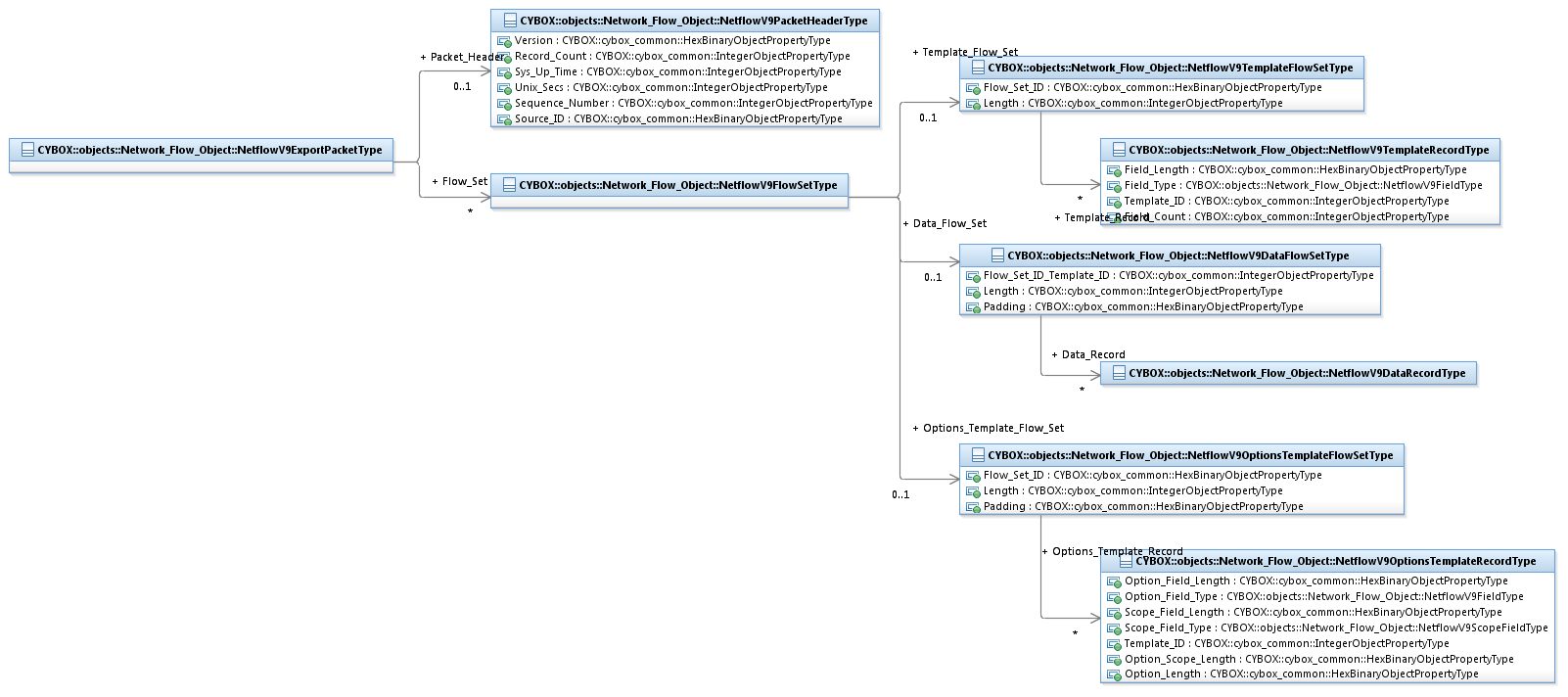


Figure 3‑3. UML diagram of the NetworkV9ExportPacketType class

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 3‑3**.

The property table of the NetflowV9ExportPacketType class is given in **Table 3‑20**.

Table 3‑20. Properties of the NetflowV9ExportPacketType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Packet\_Header** | NetflowV9PacketHeaderType | 0..1 | The Packet\_Header 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 Flow\_Set 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. |

### 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 3‑21**.

Table 3‑21. Properties of the NetflowV9PacketHeaderType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Version** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Version 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 Record\_Count 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. |

### 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: a Template FlowSet, Options Template FlowSet, and Data FlowSet.

In CybOX 2.1.1, all of the properties of the NetflowV9FlowSetType class are mutually exclusive, i.e., only one property can be populated. This restriction is based on the fact that the classes NetflowV9TemplateFlowSetType, NetflowV9OptionsTemplateFlowSetType and NetflowV9DataFlowSetType can be modelled as subclasses of the IPFIXSetType class. In future releases, it will probably be modelled that way.

The property table of the NetflowV9FlowSetType class is given in **Table 3‑22**.

Table 3‑22. Properties of the NetflowV9FlowSetType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Template\_Flow\_Set** | NetflowV9TemplateFlowSetType | 0..1 | The Template\_Flow\_Set 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.  Only one of the NetflowV9FlowSetType properties can be populated. |
| **Options\_Template\_**  **Flow\_Set** | NetflowV9OptionsTemplateFlowSetType | 0..1 | The Options\_Template\_Flow\_Set property specifies an Options Template FlowSet, which is one or more Options Template Records that have been grouped together in an Export Packet.  Only one of the NetflowV9FlowSetType properties can be populated. |
| **Data\_Flow\_Set** | NetflowV9DataFlowSetType | 0..1 | The Data\_Flow\_Set 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.  Only one of the NetflowV9FlowSetType properties can be populated. |

### NetflowV9TemplateFlowSetType Class

The NetflowV9TemplateFlowSetType class specifies the format of the Template FlowSet.

The property table of the NetflowV9TemplateFlowSetType class is given in **Table 3‑23**.

Table 3‑23. Properties of the NetflowV9TemplateFlowSetType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Flow\_Set\_ID** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Flow\_Set\_ID property specifies the FlowSet ID, which is fixed to 0 for the Template FlowSet. |
| **Length** | cyboxCommon:  IntegerObjectPropertyType | 0..1 | The Length property specifies the sum of the lengths of the FlowSet ID, the Length itself, and all Template Records within this FlowSet. |
| **Template\_Record** | NetflowV9TemplateRecordType | 0..\* | The Template\_Record property specifies the Template Record region, which includes the template ID, property count, field type, and field length. |

### 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 3‑24**.

Table 3‑24. Properties of the NetflowV9TemplateRecordType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Template\_ID** | cyboxCommon:  IntegerObjectPropertyType | 0..1 | The Template\_ID 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** | cyboxCommon:  IntegerObjectPropertyType | 0..1 | The Field\_Count property specifies the number of properties in this Template Record. |
| **Field\_Type** | NetflowV9FieldType | 0..1 | The Field\_Type 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** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Field\_Length property specifies the length of the corresponding property type, in bytes. |

### 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.

### 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 3‑25**.

Table 3‑25. Properties of the NetflowV9OptionsTemplateFlowSetType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Flow\_Set\_ID** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Flow\_Set\_ID property specifies the FlowSet ID, which is fixed to 1 for the Options Template FlowSet. |
| **Length** | cyboxCommon:  IntegerObjectPropertyType | 0..1 | The Length property specifies the total length of this FlowSet in octets, including the set header, all records, and the optional padding. |
| **Options\_Template\_**  **Record** | NetflowV9OptionsTemplateRecordType | 0..\* | The Options\_Template\_Record 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** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Padding 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 Length property includes the padding bytes. Padding SHOULD be using zeros. |

### 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 3‑26**.

Table 3‑26. Properties of the NetflowV9OptionsTemplateRecordType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Template\_ID** | cyboxCommon:  IntegerObjectPropertyType | 0..1 | The Template\_ID property specifies the template ID of this Options Template, which must be greater than 255. |
| **Option\_Scope\_Length** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Option\_Scope\_Length property specifies the length of bytes of any Scope property definition contained in the Options Template Record. |
| **Option\_Length** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Option\_Length property specifies the length of bytes of any options property definitions contained in this Options Template Record. |
| **Scope\_Field\_Type** | NetflowV9ScopeFieldType | 0..1 | The Scope\_Field\_Type 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 Scope\_Field\_Length property specifies the length (in bytes) of the Scope property as it would appear in an Options Data Record. |
| **Option\_Field\_Type** | NetflowV9FieldType | 0..1 | The Option\_Field\_Type 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 Option\_Field\_Length property specifies the length (in bytes) of the Option property. |

### 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.

### 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 3‑27**.

Table 3‑27. Properties of the NetflowV9DataFlowSetType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Flow\_Set\_ID\_Template\_ID** | cyboxCommon:  IntegerObjectPropertyType | 0..1 | The Flow\_Set\_ID\_Template\_ID property specifies the FlowSet ID, which corresponds to the Template ID from a Template Flow Set or an Options Template Flow Set. |
| **Length** | cyboxCommon:  IntegerObjectPropertyType | 0..1 | The Length property specifies the length of this FlowSet. |
| **Data\_Record** | NetflowV9DataRecordType | 0..\* | The Data\_Record 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** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Padding 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. |

### 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 3‑4.**

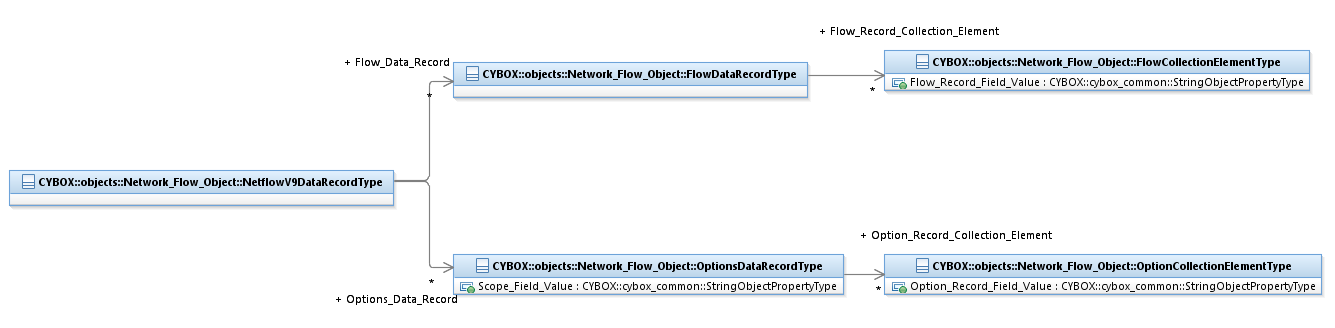


Figure 3‑4. UML diagram of the NetworkV9DataRecordType class

The property table of the NetflowV9DataRecordType class is given in **Table 3‑28**.

Table 3‑28. Properties of the NetflowV9DataRecordType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Flow\_Data\_Record** | FlowDataRecordType | 0..\* | 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. |
| **Options\_Data\_Record** | OptionsDataRecordType | 0..\* | The Options\_Data\_Record property specifies an Options Data Record, which corresponds to a previously defined Options Template Record. |

### 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 3‑29**.

Table 3‑29. Properties of the FlowDataRecordType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Flow\_Record\_Collection\_Element** | FlowCollectionElementType | 0..\* | The Flow\_Record\_Collection\_Element property specifies property values for each flow record. |

### 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 3‑30**.

Table 3‑30. Properties of the FlowCollectionElementType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Flow\_Record\_Field\_Value** | cyboxCommon:  StringObjectPropertyType | 0..\* | The Flow\_Record\_Field\_Value property specifies the set of property values for a given Flow Data Record. |

### 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 3‑31**.

Table 3‑31. 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. |

### 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 3‑32**.

Table 3‑32. 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. |

## NetflowV5PacketType Class

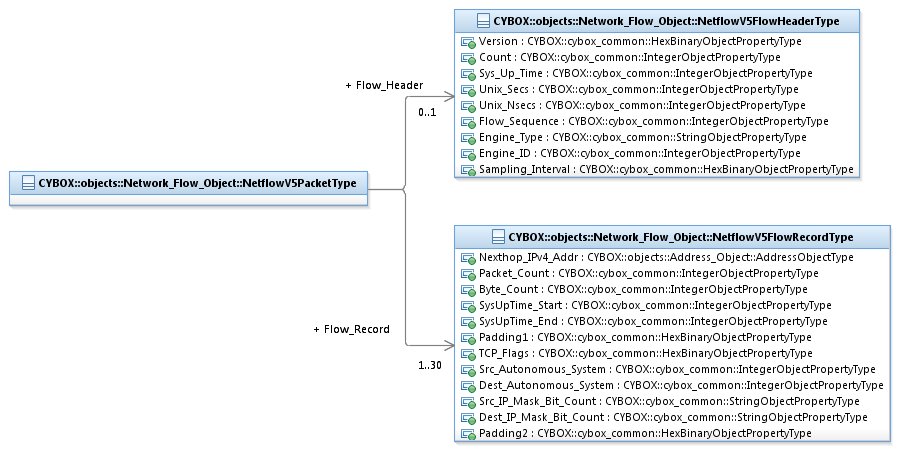


Figure 3‑5. UML diagram of the NetworkV5PacketType class

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 3‑5**.

The property table of the NetflowV5PacketType class is given in **Table 3‑33**.

Table 3‑33. Properties of the NetflowV5PacketType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Flow\_Header** | NetflowV5FlowHeaderType | 0..1 | The Flow\_Header property specifies properties of a Netflow v5 header. |
| **Flow\_Record** | NetflowV5FlowRecordType | 1..30 | The Flow\_Record 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. |

### 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 3‑34**.

Table 3‑34. Properties of the NetflowV5FlowHeaderType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Version** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Version property specifies the Netflow export format version number, which defaults to 5 in this case. |
| **Count** | cyboxCommon:  IntegerObjectPropertyType | 0..1 | The Count property specifies the number of flows exported in the packet (1-30). |
| **Sys\_Up\_Time** | cyboxCommon:  IntegerObjectPropertyType | 0..1 | The Sys\_Up\_Time property specifies the current time in milliseconds since the export device booted. |
| **Unix\_Secs** | cyboxCommon:  IntegerObjectPropertyType | 0..1 | The Unix\_Secs property specifies the current time in milliseconds since 0000 UTC 1970. |
| **Unix\_Nsecs** | cyboxCommon:  IntegerObjectPropertyType | 0..1 | The Unix\_Nsecs property specifies the residual in nanoseconds since 0000 UTC 1970. |
| **Flow\_Sequence** | cyboxCommon:  IntegerObjectPropertyType | 0..1 | The Flow\_Sequence property specifies the sequence counter of total flows seen. |
| **Engine\_Type** | cyboxCommon:  StringObjectPropertyType | 0..1 | The Engine\_Type property specifies the type of flow-switching engine. |
| **Engine\_ID** | cyboxCommon:  IntegerObjectPropertyType | 0..1 | The Engine\_ID property specifies the slot number of the flow-switching engine. |
| **Sampling\_Interval** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Sampling\_Interval property specifies the first two bits holding the sampling mode, with the remaining 14 bits holding the value of the sampling interval. |

### 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 3‑35**.

Table 3‑35. Properties of the NetflowV5FlowRecordType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Nexthop\_IPv4\_Addr** | AddressObj:  AddressObjectType | 0..1 | The Nexthop\_IPv4\_Addr property represents the IP 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. |

## SiLKRecordType Class

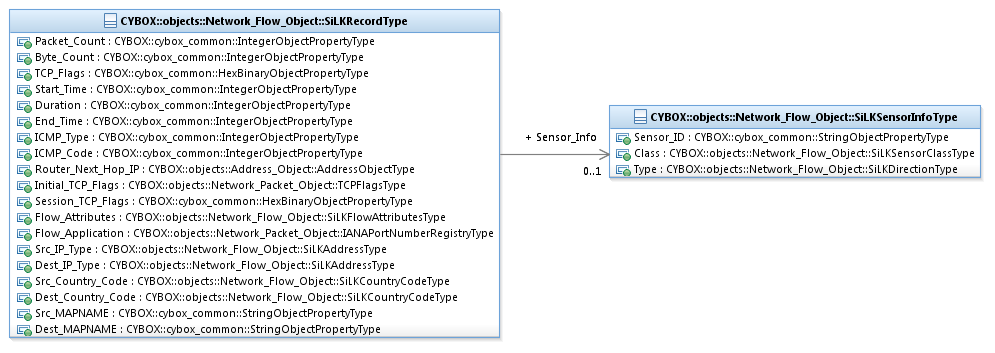


Figure 3‑6. UML diagram of the SiLKRecordType class

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](http://tools.netsa.cert.org/silk/faq.html%23ipfix-fields).

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

The property table of the SiLKRecordType class is given in **Table 3‑36**.

Table 3‑36. Properties of the SiLKRecordType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **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 number of Layer 3 bytes in the packets of the flow. |
| **TCP\_Flags** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The TCP\_Flags property specifies the union of all TCP flags observed over the life of the flow. |
| **Start\_Time** | cyboxCommon:  IntegerObjectPropertyType | 0..1 | The Start\_Time property represents the SysUpTime at start of flow, i.e. the total time in milliseconds starting from when the router booted. There is another element "Start\_Time + msec" which is the starting time of flow including milliseconds, but milliseconds are the resolution of Start\_Time unless the -legacy-timestamps switch is specified, so "Start\_Time + msec" is not defined separately. |
| **Duration** | cyboxCommon:  IntegerObjectPropertyType | 0..1 | The Duration property specifies the duration of the flow. There is another element "Duration + msec" which is the starting time of flow including milliseconds, but milliseconds are the resolution of Duration unless the -legacy-timestamps switch is specified, so "Duration + msec" is not defined separately. |
| **End\_Time** | cyboxCommon:  IntegerObjectPropertyType | 0..1 | The End\_Time property represents the SysUpTime at end of flow. There is another element "End\_Time + msec" which is the starting time of flow including milliseconds, but milliseconds are the resolution of End\_Time unless the -legacy-timestamps switch is specified, so "End\_Time + msec" is not defined separately. |
| **Sensor\_Info** | SiLKSensorInfoType | 0..1 | The Sensor\_Info property defines the properties associated with the sensor at the collection point. |
| **ICMP\_Type** | cyboxCommon:  IntegerObjectPropertyType | 0..1 | The ICMP\_Type property specifies the type for ICMP flows. It is empty for non-ICMP flows. |
| **ICMP\_Code** | cyboxCommon:  IntegerObjectPropertyType | 0..1 | The ICMP\_Code property specifies the code for ICMP flows. It is empty for non-ICMP flows. |
| **Router\_Next\_Hop\_IP** | AddressObj:  AddressObjectType | 0..1 | The Router\_Next\_Hop\_IP property specifies the router next hop IP. |
| **Initial\_TCP\_Flags** | PacketObj:TCPFlagsType | 0..1 | The Initial\_TCP\_Flags property specifies the TCP flags on first packet in the flow. |
| **Session\_TCP\_Flags** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Session\_TCP\_Flags property specifies the bit-wise OR of TCP flags over all packets except the first in the flow. |
| **Flow\_Attributes** | SiLKFlowAttributesType | 0..1 | The Flow\_Attributes property specifies the flow attributes set by the flow generator. |
| **Flow\_Application** | PacketObj:  IANAPortNumberRegistryType | 0..1 | The Flow\_Application 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 Src\_IP\_Type 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 Dest\_IP\_Type 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 Src\_Country\_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 Dest\_Country\_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 Src\_MAPNAME 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 Dest\_MAPNAME 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). |

### 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.

### 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.

### 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.

### SiLKSensorInfoType Class

The SiLKSensorInfoType class specifies properties associated with a SiLK sensor.

The property table of the SiLKSensorInfoType class is given in **Table 3‑37**.

Table 3‑37. Properties of the SiLKSensorInfoType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Sensor\_ID** | cyboxCommon:  StringObjectPropertyType | 0..1 | The Sensor\_ID property specifies the name or ID of sensor at the collection point. |
| **Class** | SiLKSensorClassType | 0..1 | The Class property specifies the sensor class. By default, the "all" class. Others can be configured. |
| **Type** | SiLKDirectionType | 0..1 | The Type property specifies the direction of traffic, which is enumerated by SiLKDirectionType. |

### 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.

### 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.

## YAFRecordType Class

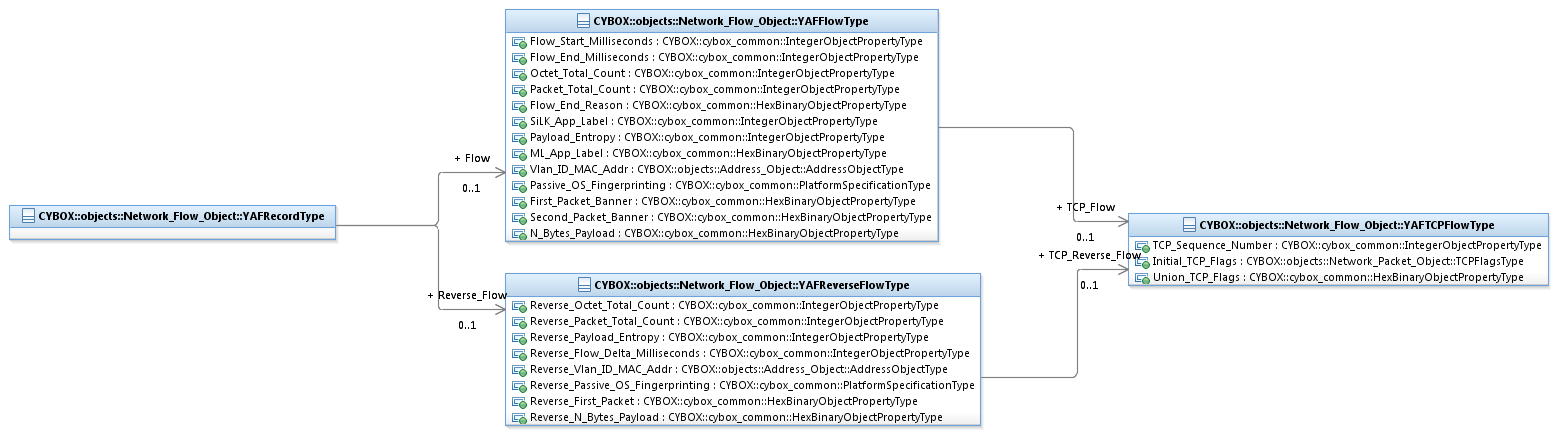


Figure 3‑7. UML diagram of the YAFRecordType class

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 3‑7**.

The property table of the YAFRecordType class is given in **Table 3‑38**.

Table 3‑38. Properties of the YAFRecordType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Flow** | YAFFlowType | 0..1 | The Flow 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 Reverse\_Flow property specifies some of the properties in a YAF record which correspond to the reverse flow. |

### 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 3‑39**.

Table 3‑39. Properties of the YAFFlowType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Flow\_Start\_Milliseconds** | cyboxCommon:  IntegerObjectPropertyType | 0..1 | The Flow\_Start\_Milliseconds property specifies the flow start time in milliseconds since 1970-01-01 00:00:00 UTC. |
| **Flow\_End\_Milliseconds** | cyboxCommon:  IntegerObjectPropertyType | 0..1 | The Flow\_End\_Milliseconds property specifies the flow end time in milliseconds since 1970-01-01 00:00:00 UTC. |
| **Octet\_Total\_Count** | cyboxCommon:  IntegerObjectPropertyType | 0..1 | The Octet\_Total\_Count 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:  IntegerObjectPropertyType | 0..1 | The Packet\_Total\_Count property specifies the number of packets in forward direction of flow. |
| **Flow\_End\_Reason** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Flow\_End\_Reason property specifies the reason for Flow termination. It may contain SiLK-specific tags. The range of values may include the following:   * 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.)   See [http://www.iana.org/assignments/ipfix/ipfix.xml](http://www.iana.org/assignments/ipfix/ipfix.xml%20) for more information. |
| **SiLK\_App\_Label** | cyboxCommon:  IntegerObjectPropertyType | 0..1 | The SiLK\_App\_Label property specifies the port number that is traditionally used for that type of traffic (see the /etc/services 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). |
| **Payload\_Entropy** | cyboxCommon:  IntegerObjectPropertyType | 0..1 | The Payload\_Entropy 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 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 ML\_App\_Label property specifies the machine-learning app label. |
| **TCP\_Flow** | YAFTCPFlowType | 0..1 | The TCP\_Flow property specifies the TCP-related information of the network flow. |
| **Vlan\_ID\_MAC\_Addr** | AddressObj:AddressObjectType | 0..1 | The Vlan\_ID\_MAC\_Addr property specifies the MAC address. |
| **Passive\_OS\_Fingerprinting** | cyboxCommon:  PlatformSpecificationType | 0..1 | The Passive\_OS\_Fingerprinting property specifies the OS name and version. |
| **First\_Packet\_Banner** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The First\_Packet\_Banner property specifies the first forward packet IP payload. |
| **Second\_Packet\_Banner** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Second\_Packet\_Banner property specifies the second forward packet IP payload. |
| **N\_Bytes\_Payload** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The N\_Bytes\_Payload property specifies the initial n bytes of forward direction of applications payload. |

### 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 3‑40**.

Table 3‑40. 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. |

### 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 3‑41**.

Table 3‑41. 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. |

## NetflowV9FieldTypeEnum Enumeration

The literals of the NetflowV9FieldTypeEnum enumeration are given in **Table 3‑42**.

Table 3‑42. 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. |

## NetflowV9ScopeFieldTypeEnum Enumeration

The literals of the NetflowV9ScopeFieldTypeEnum enumeration are given in **Table 3‑43**.

Table 3‑43. 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. |

## SiLKFlowAttributesTypeEnum Enumeration

The literals of the SiLKFlowAttributesTypeEnum enumeration are given in **Table 3‑44**.

Table 3‑44. 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 yaf(1) is run with the --silk 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). |

## SiLKAddressTypeEnum Enumeration

The literals of the SiLKAddressTypeEnum enumeration are given in **Table 3‑45**.

Table 3‑45. 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. |

## SiLKDirectionTypeEnum Enumeration

The literals of the SiLKDirectionTypeEnum enumeration are given in **Table 3‑46**.

Table 3‑46. 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. |

## SiLKSensorClassTypeEnum Enumeration

The literals of the SiLKSensorClassTypeEnum enumeration are given in **Table 3‑47**.

Table 3‑47. Literals of the SiLKSensorClassTypeEnum enumeration

|  |  |
| --- | --- |
| **Enumeration Literal** | **Description** |
| **all** | Defines sensor class "all". |

# 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.

1. 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  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  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  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 | **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  Rutger Prins  Andrei Sîrghi  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.**  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**  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.

1. Revision History

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