



CybOX™ Version 2.1.1. Part 38: Network Packet Object

Committee Specification Draft 01 / Public Review Draft 01

20 June 2016

Specification URIs

This version:

<http://docs.oasis-open.org/cti/cybox/v2.1.1/csprd01/part38-network-packet/cybox-v2.1.1-csprd01-part38-network-packet.docx> (Authoritative)
<http://docs.oasis-open.org/cti/cybox/v2.1.1/csprd01/part38-network-packet/cybox-v2.1.1-csprd01-part38-network-packet.html>
<http://docs.oasis-open.org/cti/cybox/v2.1.1/csprd01/part38-network-packet/cybox-v2.1.1-csprd01-part38-network-packet.pdf>

Previous version:

N/A

Latest version:

<http://docs.oasis-open.org/cti/cybox/v2.1.1/part38-network-packet/cybox-v2.1.1-part38-network-packet.docx> (Authoritative)
<http://docs.oasis-open.org/cti/cybox/v2.1.1/part38-network-packet/cybox-v2.1.1-part38-network-packet.html>
<http://docs.oasis-open.org/cti/cybox/v2.1.1/part38-network-packet/cybox-v2.1.1-part38-network-packet.pdf>

Technical Committee:

OASIS Cyber Threat Intelligence (CTI) TC

Chair:

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

Editors:

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

Additional artifacts:

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

Related work:

This specification is related to:

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

Abstract:

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

Status:

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

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

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

Citation format:

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

[CybOX-v2.1.1-network-packet]

CybOX™ Version 2.1.1. Part 38: Network Packet Object. Edited by Desiree Beck, Trey Darley, Ivan Kirillov, and Rich Piazza. 20 June 2016. OASIS Committee Specification Draft 01 / Public Review Draft 01. <http://docs.oasis-open.org/cti/cybox/v2.1.1/csprd01/part38-network-packet/cybox-v2.1.1-csprd01-part38-network-packet.html>. Latest version: <http://docs.oasis-open.org/cti/cybox/v2.1.1/part38-network-packet/cybox-v2.1.1-part38-network-packet.html>.

Notices

Copyright © OASIS Open 2016. All Rights Reserved.11

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

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

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

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

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

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

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

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

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

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

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

Table of Contents

1	Introduction.....	7
1.1	CybOX™ Specification Documents.....	7
1.2	Document Conventions.....	7
1.2.1	Fonts.....	7
1.2.2	UML Package References.....	8
1.2.3	UML Diagrams.....	8
1.2.4	Property Table Notation.....	9
1.2.5	Property and Class Descriptions.....	9
1.3	Terminology.....	10
1.4	Normative References.....	10
2	Background Information.....	11
2.1	Cyber Observables.....	11
2.2	Objects.....	11
3	Data Model.....	12
3.1	NetworkPacketObjectType Class.....	12
3.2	LinkLayerType Class.....	15
3.2.1	LogicalProtocolType Class.....	16
3.2.2	PhysicalInterfaceType Class.....	32
3.3	InternetLayerType Class.....	35
3.3.1	IPVersionType Data Type.....	37
3.3.2	IPv4PacketType Class.....	37
3.3.3	IPv6PacketType Class.....	42
3.3.4	ICMPv4PacketType Class.....	57
3.3.5	ICMPv6PacketType Class.....	81
3.4	TransportLayerType Class.....	94
3.4.1	TCPType Class.....	95
3.4.2	UDPType Class.....	99
3.5	IANAPortNumberRegistryType Data Type.....	100
3.6	Enumerations.....	100
3.6.1	ARPOpTypeEnum Enumeration.....	100
3.6.2	DoNotFragmentTypeEnum Enumeration.....	101
3.6.3	MoreFragmentsTypeEnum Enumeration.....	102
3.6.4	IPv4CopyFlagTypeEnum Enumeration.....	102
3.6.5	IPv4ClassTypeEnum Enumeration.....	102
3.6.6	IPv4OptionsTypeEnum Enumeration.....	103
3.6.7	IPv6DoNotRecogActionTypeEnum Enumeration.....	105
3.6.8	IPv6PacketChangeTypeEnum Enumeration.....	106
3.6.9	IPVersionTypeEnum Enumeration.....	106
3.6.10	IANAHardwareTypeEnum Enumeration.....	107
3.6.11	IANAEtherTypeEnum Enumeration.....	108
3.6.12	IANAAssignedIPNumbersTypeEnum Enumeration.....	109
3.6.13	IANAPortNumberRegistryTypeEnum Enumeration.....	111
3.6.14	MFlagTypeEnum Enumeration.....	112

4	Conformance	113
	Appendix A. Acknowledgments	114
	Appendix B. Revision History	118

1 Introduction

[All text is normative unless otherwise labeled]

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

This document serves as the specification for the CybOX Network Packet 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 Packet Object data model. We present the Network Packet Object data model specification details in Section 3 and conformance information in Section 4.

1.1 CybOX™ Specification Documents

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

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

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

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

1.2 Document Conventions

The following conventions are used in this document.

1.2.1 Fonts

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

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

Examples: Action, Object, Event, Property

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

Examples: ActionType, cyboxCommon:BaseObjectPropertyType

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

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

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

1.2.2 UML Package References

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

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

1.2.3 UML Diagrams

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

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

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

1.2.3.1 Class Properties

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

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

1.2.3.2 Diagram Icons and Arrow Types

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

Table 1-1. UML diagram icons

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

1.2.4 Property Table Notation

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

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

1.2.5 Property and Class Descriptions

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

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

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

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

1.3 Terminology

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

1.4 Normative References

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

2 Background Information

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

2.1 Cyber Observables

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

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

2.2 Objects

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

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

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

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

3 Data Model

3.1 NetworkPacketObjectType Class

The `NetworkPacketObjectType` class is based on the TCP/IP model/Internet protocol suite. In the TCP/IP stack, a "packet" is generally defined as IP header plus payload, but we also include the Link Layer, which defines the physical network interfaces and routing protocols and the Transport Layer, which defines provide host-to-host communication services for applications. Protocol fields are provided but requirements are not enforced/captured.

See <https://tools.ietf.org/html/rfc1122> for more information.

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

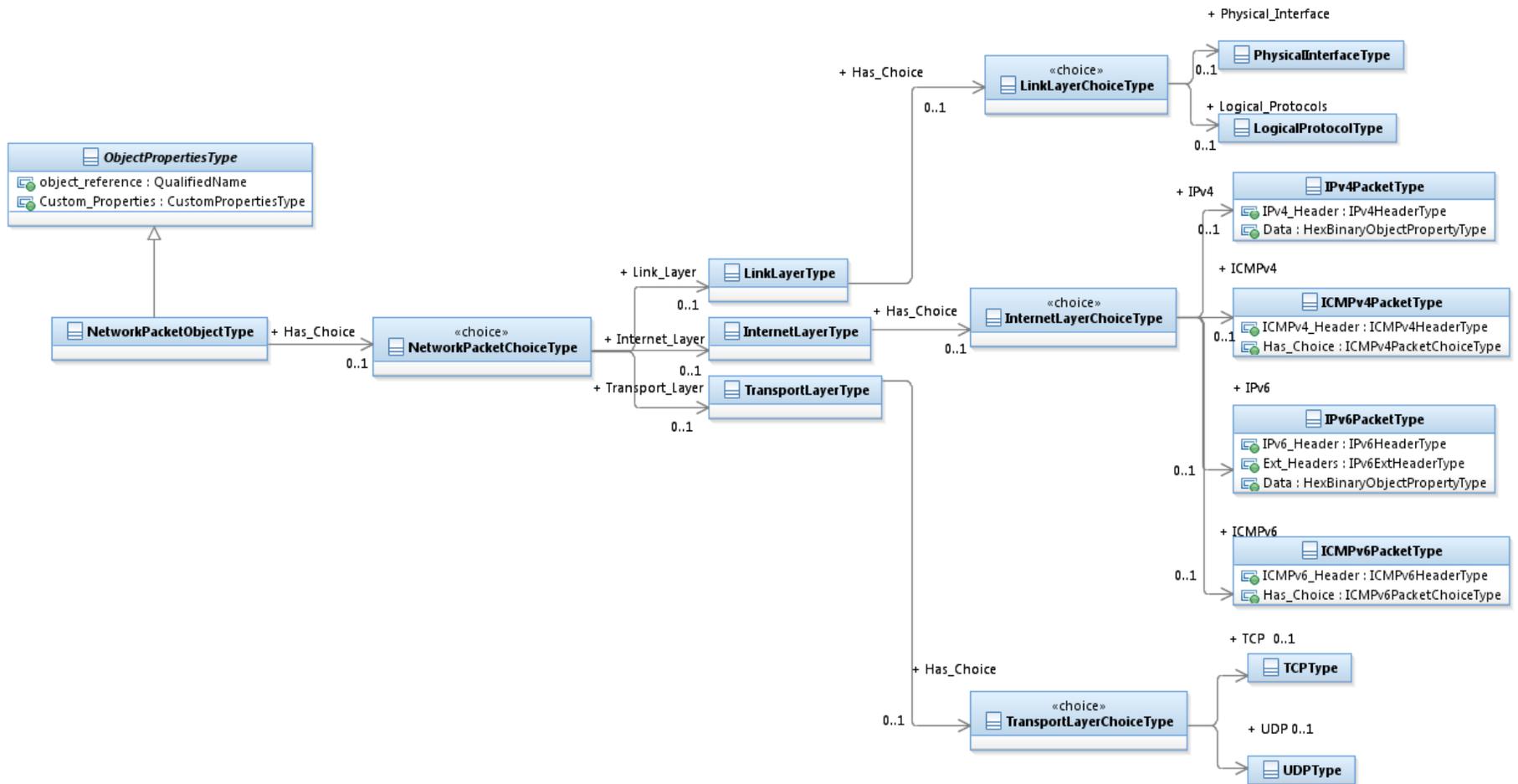


Figure 3-1. UML diagram of the NetworkPacketObjectType class¹

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

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

Name	Type	Multiplicity	Description
Has_Choice	<code>NetworkPacketObjectChoiceType</code>	1	<p>The <code>Has_Choice</code> property is associated with the class <code>NetworkPacketObjectChoiceType</code>. It indicates that there is a choice among the <code>Link_Layer</code>, <code>Internet_Layer</code>, and <code>Transport_Layer</code> properties.</p> <p>Only one of the properties of <code>NetworkPacketObjectChoiceType</code> class can be populated at any time. See Section 3.1 for more detail.</p>

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

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

Name	Type	Multiplicity	Description
Link_Layer	<code>LinkLayerType</code>	0..1	<p>The <code>Link_Layer</code> property is the lowest layer of the TCP/IP network stack and is comprised of physical and logical protocols that operate between adjacent nodes of a network segment or a WAN connection.</p> <p>Only one of the properties of <code>NetworkPacketObjectChoiceType</code> can be populated.</p>
Internet_Layer	<code>InternetLayerType</code>	0..1	<p>The <code>Internet_Layer</code> property characterizes information about the network layer of this packet, as defined in the 7-layer OSI model. See https://en.wikipedia.org/wiki/OSI_model for more details.</p> <p>Only one of the properties of <code>NetworkPacketObjectChoiceType</code> can be</p>

			populated.
Transport_Layer	TransportLayerType	0..1	<p>The <code>Transport_Layer</code> property characterizes information about the transport layer of this packet as defined in the 7-layer OSI model. See https://en.wikipedia.org/wiki/OSI_model for more details.</p> <p>Only one of the properties of <code>NetworkPacketObjectChoiceType</code> can be populated.</p>

3.2 LinkLayerType Class

The `LinkLayerType` class characterizes the link layer protocol, which is a hardware interface protocol, such as Ethernet, or a logical link routing protocol, such as ARP. The UML diagram corresponding to the `LinkLayerObjectType` class is shown in [Figure 3-1](#).

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

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

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

The `LinkLayerChoiceType` class is the type of the `Has_Choice` property. In the UML model, this class is associated with the `<<choice>>` UML stereotype, which specifies that only one of the available properties of the `LinkLayerChoiceType` class can be populated at any time. The property table of the `LinkLayerChoiceType` class is given in [Table 3-4](#).

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

Name	Type	Multiplicity	Description
Physical_Interface	PhysicalInterfaceType	0..1	The <code>Physical_Interface</code> property characterizes one hardware interface of a link layer connection. Only one of the properties of <code>LinkLayerChoiceType</code> can be populated.
Logical_Protocols	LogicalProtocolType	0..1	The <code>Logical_Protocols</code> property characterizes the logical protocol of a link layer connection. One example of a logical protocol is ARP. Only one of the properties of <code>LinkLayerChoiceType</code> can be populated.

3.2.1 LogicalProtocolType Class

The `LogicalProtocolType` class characterizes the logical protocol of a link layer connection. One example of a logical protocol is ARP.

The UML diagram corresponding to the `LogicalProtocolType` class is shown in [Figure 3-2](#).



Figure 3-2. UML diagram for the `LogicalProtocolType` class

The property table of the `LogicalProtocolType` class is given in [Table 3-5](#).

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

Name	Type	Multiplicity	Description
Has_Choice	LogicalProtocolChoiceType	0..1	<p>The <code>Has_Choice</code> property is associated with the class <code>LogicalProtocolChoiceType</code>. It indicates that there is a choice among the <code>ARP_RARP</code> property and the <code>NDP</code> property.</p> <p>Only one of the properties of <code>LogicalProtocolChoiceType</code> class can be populated at any time. See Section 3.2.1 for more detail.</p>

The `LogicalProtocolChoiceType` class is the type of the `Has_Choice` property. In the UML model, this class is associated with the `<<choice>>` UML stereotype, which specifies that only one of the available properties of the `LogicalProtocolChoiceType` class can be populated at any time. The property table of the `LogicalProtocolChoiceType` class is given in Table 3-6.

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

Name	Type	Multiplicity	Description
ARP_RARP	ARPTYPE	0..1	<p>The <code>ARP_RARP</code> property specifies either the ARP or the RARP logical protocol. ARP is a logical protocol used for resolution of network layer addresses (e.g., IP addresses) into link layer addresses (e.g., MAC addresses). RARP is a logical protocol used by a host computer to request its network layer address when it has its link layer address.</p> <p>Only one of the properties of <code>LogicalProtocolChoiceType</code> can be populated.</p>
NDP	NDPTYPE	0..1	<p>The <code>NDP</code> property specifies the Neighbor Discovery Protocol, which is used with IPv6 to determine the link-layer addresses for neighbors. It corresponds to a combination of IPv4 protocols: ARP, ICMP Router Discovery, and ICMP Redirect.</p> <p>Only one of the properties of <code>LogicalProtocolChoiceType</code> can be populated.</p>

3.2.1.1 ARPTYPE Class

The ARPTYPE class characterizes the Address Resolution Protocol, which is a request and reply protocol that runs encapsulated by the line protocol. It is communicated within the boundaries of a single network, and never routed across inter-network nodes. This property places ARP into the Link Layer. See <http://www.comptechdoc.org/independent/networking/guide/netarp.html> for more information.

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

Table 3-7. Properties of the ARPTYPE class

Name	Type	Multiplicity	Description
Hardware_Addr_Type	IANAHardwareType	0..1	The Hardware_Addr_Type property characterizes the type of hardware address specified in an ARP message.
Proto_Addr_Type	IANAEtherType	0..1	The Proto_Addr_Type property characterizes the type of protocol address being mapped. For IPv4 addresses, the value is 0x0800.
Hardware_Addr_Size	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Hardware_Addr_Size property represents the byte size of the hardware address. For Ethernet or other IEEE 802 MAC addresses, the value is 6.
Proto_Addr_Size	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Proto_Addr_Size property represents the byte size of the protocol address. For IPv4 addresses, the value is 4.
Op_Type	ARPOpType	0..1	The Op_Type property characterizes the type of operation. 1 for an ARP request, 2 for a ARP reply, 3 for a RARP request and 4 for a RARP reply.
Sender_Hardware_Addr	AddressObj: AddressObjectType	0..1	The Sender_Hardware_Addr property characterizes the sender's hardware address (e.g., MAC address).
Sender_Protocol_Addr	AddressObj:	0..1	The Sender_Protocol_Addr property characterizes the

	AddressObjectType		sender's IP address.
Recip_Hardware_Addr	AddressObj: AddressObjectType	0..1	The Recip_Hardware_Addr property characterizes the recipients' hardware address (e.g., MAC address).
Recip_Protocol_Addr	AddressObj: AddressObjectType	0..1	The Recip_Protocol_Addr property characterizes the recipient's IP address.

3.2.1.1.1 IANAHardwareType Data Type

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

3.2.1.1.2 IANAEtherType Data Type

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

3.2.1.1.3 ARPOpType Data Type

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

3.2.1.2 NDPTYPE Class

The `NDPTYPE` class characterizes a Neighbor Discover Protocol (NDP) IPv6 packet. There are five different types of NDP packets. See <http://tools.ietf.org/html/rfc4861> for more information.

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

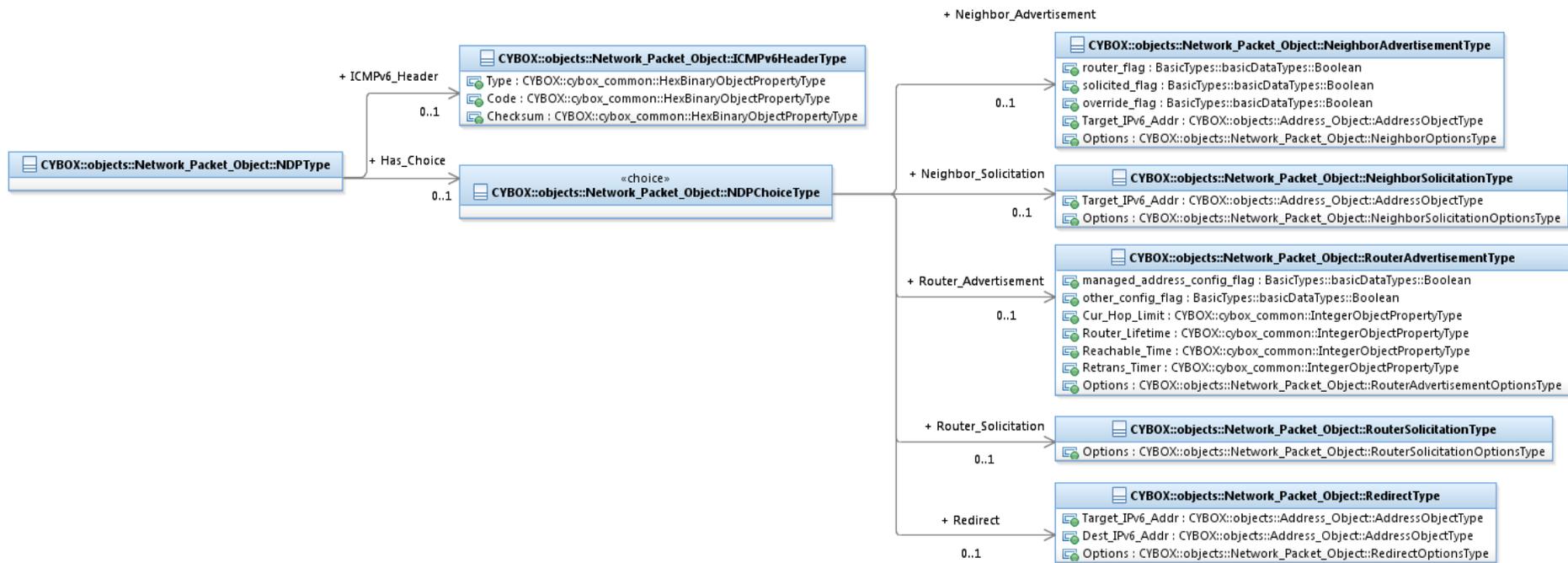


Figure 3-3. UML diagram for the NDPTYPE class

The property table of the NDPTYPE class is given in Table 3-8.

Table 3-8. Properties of the NDPTYPE class

Name	Type	Multiplicity	Description
Has_Choice	NDPTYPEChoiceType	0..1	The Has_Choice property is associated with the class NDPTYPEChoiceType. It indicates that there is a choice among the Router_Solicitation, Router_Advertisement, Neighbor_Solicitation, Neighbor_Advertisement and Redirect properties.

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

The `NDPChoiceType` class is the type of the `Has_Choice` property. In the UML model, this class is associated with the `<<choice>>` UML stereotype, which specifies that only one of the available properties of the `NDPChoiceType` class can be populated at any time. The property table of the `NDPChoiceType` class is given in [Table 3-9](#).

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

Name	Type	Multiplicity	Description
ICMPv6_Header	<code>ICMPv6HeaderType</code>	0..1	The <code>ICMPv6_Header</code> property characterizes an ICMPv6 header.
Router_Solicitation	<code>RouterSolicitationType</code>	0..1	<p>The <code>Router_Solicitation</code> property is used to specify a <code>RouterSolicitationType</code> form of the <code>NDPType</code> class. Hosts send Router Solicitations in order to prompt routers to generate Router Advertisements quickly (type=133; code=0).</p> <p>Only one of the <code>Router_Solicitation</code>, <code>Router_Advertisement</code>, <code>Neighbor_Solicitation</code>, <code>Neighbor_Advertisement</code> and <code>Redirect</code> properties can be populated.</p>
Router_Advertisement	<code>RouterAdvertisementType</code>	0..1	<p>The <code>Router_Advertisement</code> property is used to specify a <code>RouterAdvertisementType</code> form of the <code>NDPType</code> class. Routers send out Router Advertisement messages periodically, or in response to Router Solicitations (type=134; code=0).</p> <p>Only one of the <code>Router_Solicitation</code>, <code>Router_Advertisement</code>, <code>Neighbor_Solicitation</code>, <code>Neighbor_Advertisement</code> and <code>Redirect</code> properties can be populated.</p>

Neighbor_Solicitation	NeighborSolicitationType	0..1	<p>The Neighbor_Solicitation property is used to specify a NeighborSolicitationType form of the NDPTYPE class. Nodes send Neighbor Solicitations to request the link-layer address of a target node while also providing their own link-layer address to the target. Neighbor Solicitations are multicast when the node needs to resolve an address and unicast when the node seeks to verify the reachability of a neighbor (type=135; code=0).</p> <p>Only one of the Router_Solicitation, Router_Advertisement, Neighbor_Solicitation, Neighbor_Advertisement and Redirect properties can be populated.</p>
Neighbor_Advertisement	NeighborAdvertisementType	0..1	<p>The Neighbor_Advertisement property is used to specify a NeighborAdvertisementType form of the NDPTYPE class. A node sends Neighbor Advertisements in response to Neighbor Solicitations and sends unsolicited Neighbor Advertisements in order to (unreliably) propagate new information quickly (type=136; code=0).</p> <p>Only one of the Router_Solicitation, Router_Advertisement, Neighbor_Solicitation, Neighbor_Advertisement and Redirect properties can be populated.</p>
Redirect	RedirectType	0..1	<p>The Redirect property is used to specify a RedirectType form of the NDPTYPE class. Routers send Redirect packets to inform a host of a better first-hop node on the path to a destination. Hosts can be redirected to a better first-hop router but can also be informed by a redirect that the destination is in fact a neighbor. The latter is accomplished by setting the ICMP Target Address equal to the ICMP Destination Address (type=137; code=0).</p>

			Only one of the Router_Solicitation, Router_Advertisement, Neighbor_Solicitation, Neighbor_Advertisement and Redirect properties can be populated.
--	--	--	--

3.2.1.2.1 RouterSolicitationType Class

The RouterSolicitationType class specifies a NDP packet which hosts send in order to prompt routers to generate Router Advertisements quickly (class=133; code=0).

The property table of the RouterSolicitationType class is given in [Table 3-10](#).

Table 3-10. Properties of the RouterSolicitationType class

Name	Type	Multiplicity	Description
Options	RouterSolicitationOptionsType	0..*	The Options property - Router Solicitation messages include zero or more options, some of which may appear multiple times in the same message.

3.2.1.2.2 RouterSolicitationOptionsType Class

The RouterSolicitationOptionsType class specifies zero or more options for a Router Solicitation packet, some of which may appear multiple times in the same message.

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

Table 3-11. Properties of the RouterSolicitationOptionsType class

Name	Type	Multiplicity	Description
Src_Link_Addr	NDPLinkAddrType	0..1	The Src_Link_Addr property characterizes the Source Link-Layer Address option.

3.2.1.2.3 RouterAdvertisementType Class

The RouterAdvertisementType class specifies a NDP packet which routers send periodically, or in response to Router Solicitations (class=134; code=0).

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

Table 3-12. Properties of the RouterAdvertisementType class

Name	Type	Multiplicity	Description
managed_address_config_flag	basicDataTypes:Boolean	0..1	The managed_address_config_flag property specifies the 1-bit "Managed address configuration" flag. When set, it indicates that addresses are available via Dynamic Host Configuration Protocol. If the M flag is set, the O flag is redundant and can be ignored because DHCPv6 will return all available configuration information.
other_config_flag	basicDataTypes:Boolean	0..1	The other_config_flag property specifies the 1-bit "Other configuration" flag. When set, it indicates that other configuration information is available via DHCPv6. Examples of such information are DNS-related information or information on other servers within the network.
Cur_Hop_Limit	cyboxCommon: IntegerObjectPropertyType	0..1	The Cur_Hop_Limit property is an 8-bit unsigned integer, which specifies the current hop limit. The default value that should be placed in the Hop Count property of the IP header for outgoing IP packets. A value of zero means unspecified (by this router).
Router_Lifetime	cyboxCommon: IntegerObjectPropertyType	0..1	The Router_Lifetime property is a 16-bit unsigned integer which specifies the lifetime associated with the default router, in units of seconds. The property can contain values up to 65535 and receivers should handle any value, while the sending rules in Section 6 limit the lifetime to 9000 seconds.

Reachable_Time	cyboxCommon: IntegerObjectPropertyType	0..1	The <code>Reachable_Time</code> property is a 32-bit unsigned integer which specifies the time, in milliseconds, between retransmitted Neighbor Solicitation messages. Used by address resolution and the Neighbor Unreachability Detection algorithm. A value of zero means unspecified (by this router).
Retrans_Timer	cyboxCommon: IntegerObjectPropertyType	0..1	The <code>Retrans_Timer</code> property is a 32-bit unsigned integer which specifies the time, in milliseconds, between retransmitted Neighbor Solicitation messages. Used by address resolution and the Neighbor Unreachability Detection algorithm. A value of zero means unspecified (by this router).
Options	RouterAdvertisementOptionsType	0..1	The <code>Options</code> property specifies zero or more options, some of which may appear multiple times in the same message.

3.2.1.2.4 RouterAdvertisementOptionsType Class

The `RouterAdvertisementOptionsType` class specifies zero or more options of a Router Advertisement packet, some of which may appear multiple times in the same message.

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

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

Name	Type	Multiplicity	Description
Src_Link_Addr	NDPLinkAddrType	0..1	The <code>Src_Link_Addr</code> property characterizes the Source Link-Layer Address option.
MTU	NDPMTUType	0..1	The <code>MTU</code> property is a 32-bit unsigned integer, which specifies the recommended MTU for the link.
Prefix_Info	NDPPrefixInfoType	0..1	The <code>Prefix_Info</code> property characterizes the Prefix Information for the Router Advertisement Options.

3.2.1.2.5 NDPLinkAddrType Class

The NDPLinkAddrType class characterizes the Link-Layer Address option.

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

Table 3-14. Properties of the NDPLinkAddrType class

Name	Type	Multiplicity	Description
Length	cyboxCommon: IntegerObjectPropertyType	0..1	The Length property of the option (including the type and length properties) in units of 8 octets.
Link_Layer_MAC_Addr	AddressObj: AddressObjectType	0..1	The Link_Layer_MAC_Addr property specifies a variable length link-layer address. The content and format of this property (including byte and bit ordering) is expected to be specified in specific documents that describe how IPv6 operates over different link layers.

3.2.1.2.6 NDPMTUType Class

The NDPMTUType class specifies the MTU option which is used in Router Advertisement messages to ensure that all nodes on a link use the same MTU value in those cases where the link MTU is not well known (class=5).

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

Table 3-15. Properties of the NDPMTUType class

Name	Type	Multiplicity	Description
Length	cyboxCommon: IntegerObjectPropertyType	0..1	The Length property specifies the length of the MTU option type: length=1.
MTU	cyboxCommon:	0..1	The MTU property is a 32-bit unsigned integer that specifies the recommended MTU for the link.

	IntegerObjectPropertyType		
--	---------------------------	--	--

3.2.1.2.7 NDPPrefixInfoType Class

The `NDPPrefixInfoType` class characterizes the prefix information for Router Advertisement Options. It provides hosts with on-link prefixes and prefixes for Address Autoconfiguration (class=3). See <http://tools.ietf.org/html/rfc4861> for more information.

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

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

Name	Type	Multiplicity	Description
link_flag	<code>basicDataTypes:Boolean</code>	0..1	The <code>link_flag</code> property specifies the 1-bit on-link flag. When set, it indicates that this prefix can be used for on-link determination. When not set the advertisement makes no statement about on-link or off-link properties of the prefix.
addr_config_flag	<code>basicDataTypes:Boolean</code>	0..1	The <code>addr_config_flag</code> property specifies the 1-bit autonomous address-configuration flag. When set, it indicates that this prefix can be used for stateless address configuration.
Length	<code>cyboxCommon:IntegerObjectPropertyType</code>	0..1	The <code>Length</code> property characterizes the length of the option (the number of valid leading bits in the prefix), and is represented as a 32-bit integer.
Prefix_Length	<code>cyboxCommon:IntegerObjectPropertyType</code>	0..1	The <code>Prefix_Length</code> property contains an 8-bit unsigned integer specifying the number of leading bits in the Prefix that are valid. The value ranges from 0 to 128. The prefix length property provides necessary information for on-link determination (when combined with the L flag in the prefix information option).
Valid_Lifetime	<code>cyboxCommon:IntegerObjectPropertyType</code>	0..1	The <code>Valid_Lifetime</code> property contains a 32-bit unsigned integer specifying the length of time in seconds (relative to the time the packet is sent) that the prefix is valid for the purpose of on-link determination.

			A value of all one bits (0xffffffff) represents infinity.
Preferred_Lifetime	cyboxCommon: IntegerObjectPropertyType	0..1	The Preferred_Lifetime property contains a 32-bit unsigned integer, specifying the length of time in seconds (relative to the time the packet is sent) that addresses generated from the prefix via stateless address autoconfiguration remain preferred.
Prefix	PrefixType	0..1	The Prefix property specifies an IP address or a prefix of an IP address for IPv6.

3.2.1.2.8 PrefixType Class

The PrefixType class characterizes an IP address or a prefix of an IP address for IPv6.

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

Table 3-17. Properties of the PrefixType class

Name	Type	Multiplicity	Description
IPv6_Addr	AddressObj:AddressObjectType	0..1	The IPv6_Addr property specifies the IPv6 address.
IP_Addr_Prefix	AddressObj:AddressObjectType	0..1	The IP_Addr_Prefix property specifies the initial bits of an IPv6 address (these are identical for all hosts in a network) from the network's prefix. See http://ipv6.com/articles/general/IPv6-Addressing.htm for more information.

3.2.1.2.9 NeighborSolicitationType Class

The NeighborSolicitationType class specifies a NDP packet which nodes send to request the link-layer address of a target node while also providing their own link-layer address to the target. Neighbor Solicitations are multicast when the node needs to resolve an address and unicast when the node seeks to verify the reachability of a neighbor (class=135; code=0).

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

Table 3-18. Properties of the NeighborSolicitationType class

Name	Type	Multiplicity	Description
Target_IPv6_Addr	AddressObj:AddressObjectType	0..1	The Target_IPv6_Addr property specifies the IP address of the target of the solicitation.
Options	NeighborSolicitationOptionsType	0..1	The Options property specifies zero or more options, some of which may appear multiple times in the same message.

3.2.1.2.10 NeighborSolicitationOptionsType Class

The NeighborSolicitationOptionsType class specifies zero or more options of a Neighbor Solicitation packet, some of which may appear multiple times in the same message.

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

Table 3-19. Properties of the NeighborSolicitationOptionsType class

Name	Type	Multiplicity	Description
Src_Link_Addr	NDPLinkAddrType	0..1	The Src_Link_Addr property characterizes the Source Link-Layer Address option.

3.2.1.2.11 NeighborAdvertisementType Class

The NeighborAdvertisementType class specifies a NDP packet which a node sends in response to Neighbor Solicitations and sends unsolicited Neighbor Advertisements in order to (unreliably) propagate new information quickly (class=136; code=0).

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

Table 3-20. Properties of the NeighborAdvertisementType class

Name	Type	Multiplicity	Description
router_flag	basicDataTypes: Boolean	0..1	The <code>router_flag</code> property specifies the router flag (R-bit). When set, the R-bit indicates that the sender is a router. The R-bit is used by Neighbor Unreachability Detection to detect a router that changes to a host.
solicited_flag	basicDataTypes: Boolean	0..1	The <code>solicited_flag</code> property specifies the solicited flag (S-bit). When set, the S-bit indicates that the advertisement was sent in response to a Neighbor Solicitation from the Destination address. The S-bit is used as a reachability confirmation for Neighbor Unreachability Detection.
override_flag	basicDataTypes: Boolean	0..1	The <code>override_flag</code> property specifies the override flag (O-bit). When set, the O-bit indicates that the advertisement should override an existing cache entry and update the cached link-layer address.
Target_IPv6_Addr	AddressObj: AddressObjectType	0..1	The <code>Target_IPv6_Addr</code> property specifies the IP address of the target of the advertisement.
Options	NeighborOptionsType	0..1	The <code>Options</code> property specifies zero or more options, some of which may appear multiple times in the same message.

3.2.1.2.12 NeighborOptionsType Class

The `NeighborOptionsType` class specifies zero or more options of a Neighbor Advertisement packet, some of which may appear multiple times in the same message.

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

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

Name	Type	Multiplicity	Description
Target_Link_Addr	NDPLinkAddrType	0..1	The <code>Target_Link_Addr</code> property characterizes the Target Link-

			Layer Address option.
--	--	--	-----------------------

3.2.1.2.13 RedirectType Class

The `RedirectType` class specifies a NDP packet which Routers send to inform a host of a better first-hop node on the path to a destination. Hosts can be redirected to a better first-hop router but can also be informed by a redirect that the destination is in fact a neighbor. The latter is accomplished by setting the ICMP Target Address equal to the ICMP Destination Address (class=137; code=0).

The property table of the `RedirectType` class is given in [Table 3-22](#).

Table 3-22. Properties of the `RedirectType` class

Name	Type	Multiplicity	Description
Target_IPv6_Addr	AddressObj: AddressObjectType	0..1	The <code>Target_IPv6_Addr</code> property specifies an IP address that is a better first hop to use for the ICMP Destination Address.
Dest_IPv6_Addr	AddressObj: AddressObjectType	0..1	The <code>Dest_IPv6_Addr</code> property specifies the IP address of the destination that is redirected to the target.
Options	RedirectOptionsType	0..1	The <code>Options</code> property specifies zero or more options, some of which may appear multiple times in the same message.

3.2.1.2.14 RedirectOptionsType Class

The `RedirectOptionsType` class specifies zero or more options of a Redirect packet, some of which may appear multiple times in the same message.

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

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

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

Target_Link_Addr	NDPLinkAddrType	0..1	The Target_Link_Addr property specifies the link-layer address for the target.
Redirected_Header	NDPRedirectedHeaderType	0..1	The Redirected_Header property specifies as much of the IP packet as possible that triggered the sending of the Redirect message without making the redirect packet exceed the minimum MTU specified in the IPv6 protocol.

3.2.1.2.15 NDPRedirectedHeaderType Class

The NDPRedirectedHeaderType class is used in redirect packets and contains all or part of the packet that is being redirected (class=4).

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

Table 3-24. Properties of the NDPRedirectedHeaderType class

Name	Type	Multiplicity	Description
Length	cyboxCommon: IntegerObjectPropertyType	0..1	The Length property specifies the length of the option (including the type and length properties) in units of 8 octets.
IPHeader_And_Data	cyboxCommon: HexBinaryObjectPropertyType	0..1	The IPHeader_And_Data property specifies as much of the IP packet that triggered the sending of the redirect without making redirect packet larger than MTU.

3.2.2 PhysicalInterfaceType Class

The PhysicalInterfaceType class specifies the interface to the physical network. Multiple interface classes exist, however, only the most common (Ethernet) is defined. Others will be added later as needed.

The UML diagram corresponding to the PhysicalInterfaceType class is shown in [Figure 3-4](#).

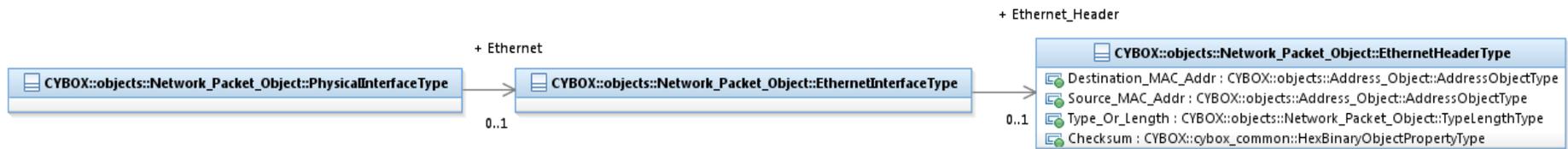


Figure 3-4. UML diagram for the *PhysicalInterfaceType* class

The property table of the *PhysicalInterfaceType* class is given in [Table 3-25](#).

Table 3-25. Properties of the *PhysicalInterfaceType* class

Name	Type	Multiplicity	Description
Ethernet	<i>EthernetInterfaceType</i>	0..1	The <i>Ethernet</i> property specifies the Ethernet interface that sends network packets from the sending host to one or more receiving hosts. See http://www.ieee802.org/3/ and http://wiki.wireshark.org/Ethernet for more information.

3.2.2.1 EthernetInterfaceType Class

The *EthernetInterfaceType* class characterizes the Ethernet interface, which is used to send network packets from the sending host to one or more receiving hosts.

See <http://www.ieee802.org/3/> and <http://wiki.wireshark.org/Ethernet> for more information.

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

Table 3-26. Properties of the *EthernetInterfaceType* class

Name	Type	Multiplicity	Description
Ethernet_Header	<i>EthernetHeaderType</i>	0..1	The <i>Ethernet_Header</i> property includes information such as source MAC address, destination MAC address, and more.

3.2.2.1.1 EthernetHeaderType Class

The `EthernetHeaderType` class characterizes the Ethernet header and includes information such as source MAC address, destination MAC address, and more.

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

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

Name	Type	Multiplicity	Description
Destination_MAC_Addr	AddressObj: AddressObjectType	0..1	The <code>Destination_MAC_Addr</code> property characterizes the destination MAC Address of the Ethernet frame.
Source_MAC_Addr	AddressObj: AddressObjectType	0..1	The <code>Source_MAC_Addr</code> property characterizes the source MAC Address of the Ethernet frame.
Type_Or_Length	TypeLengthType	0..1	The <code>Type_Or_Length</code> property characterizes either the length of the Ethernet frame or the protocol type of the network layer.
Checksum	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>Checksum</code> property characterizes the Frame Check sequence of an Ethernet frame.

3.2.2.1.2 TypeLengthType Class

The `TypeLengthType` class can specify either the length or the protocol type. If the value is 0-1500, then a length is being specified. Otherwise, it specifies the protocol class of the Internet layer.

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

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

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

Length	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Length property characterizes the length of the Ethernet frame.
Internet_Layer_Type	IANAEtherType	0..1	The Internet_Layer_Type property consists of two octets in an Ethernet frame, and specifies the protocol encapsulated in the payload of frame.

3.3 InternetLayerType Class

The `InternetLayerType` class specifies the group of methods, protocols, and specifications that are used to transport packets from the originating host across network boundaries. Only protocols most commonly used are currently defined: IPv4, ICMPv4, IPv6 and ICMPv6. Other protocols will be added as needed. See http://en.wikipedia.org/wiki/Internet_layer for more information.

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

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

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

Name	Type	Multiplicity	Description
Has_Choice	<code>InternetLayerChoiceType</code>	1	<p>The <code>Has_Choice</code> property is associated with the class <code>InternetLayerChoiceType</code>. It indicates that there is a choice among IPv4, ICMPv4, IPv6 and ICMPv6 properties.</p> <p>Only one of the properties of <code>InternetLayerChoiceType</code> class can be populated at any time. See Section 1.2.3 for more detail.</p>

The `InternetLayerChoiceType` class is the type of the `Has_Choice` property. In the UML model, this class is associated with the `<<choice>>` UML stereotype, which specifies that only one of the available properties of the `InternetLayerChoiceType` class can be populated at any time. The property table of the `InternetLayerChoiceType` class is given in [Table 3-30](#).

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

Name	Type	Multiplicity	Description
IPv4	IPv4PacketType	0..1	<p>The <code>IPv4</code> property specifies the Internet Protocol version 4 (IPv4) packet. IPv4 is a connectionless protocol for use on packet-switched link layer networks (e.g., Ethernet).</p> <p>Only one of the properties of <code>InternetLayerChoiceType</code> can be populated.</p>
ICMPv4	ICMPv4PacketType	0..1	<p>The <code>ICMPv4</code> property specifies an ICMP packet (v4) which is chiefly used in the operating systems of networked computers to send error messages indicating, for example, that a requested service is not available or that a host or router could not be reached. See http://en.wikipedia.org/wiki/Internet_Control_Message_Protocol and http://www.networksorcery.com/enp/protocol/icmp.htm for more information.</p> <p>Only one of the properties of <code>InternetLayerChoiceType</code> can be populated.</p>
IPv6	IPv6PacketType	0..1	<p>The <code>IPv6</code> property specifies the Internet Protocol version 6 (IPv6) packets. IPv6 is a connectionless protocol for use on packet-switched link layer networks which is intended to succeed IPv4.</p> <p>Only one of the properties of <code>InternetLayerChoiceType</code> can be populated.</p>
ICMPv6	ICMPv6PacketType	0..1	<p>The <code>ICMPv6</code> property specifies an ICMP packet (v6). ICMPv6 performs error reporting and diagnostic functions.</p> <p>Only one of the properties of <code>InternetLayerChoiceType</code> can be populated.</p>

3.3.1 IPVersionType Data Type

The `IPVersionType` data type specifies the version of the IP protocol is being used. Its core value SHOULD be a literal found in the `IPVersionTypeEnum` enumeration. Its base type is the `BaseObjectPropertyType` data type, in order to permit complex (i.e. regular-expression based) specifications.

3.3.2 IPv4PacketType Class

The `IPv4PacketType` class is used to characterize a packet using the Internet Protocol version 4 (IPv4), which is a connectionless protocol for use on packet-switched link layer networks (e.g., Ethernet). See <http://tools.ietf.org/html/rfc791> and <http://en.wikipedia.org/wiki/IPv4> for more information.

The UML diagram corresponding to the `IPv4PacketType` class is shown in [Figure 3-5](#).

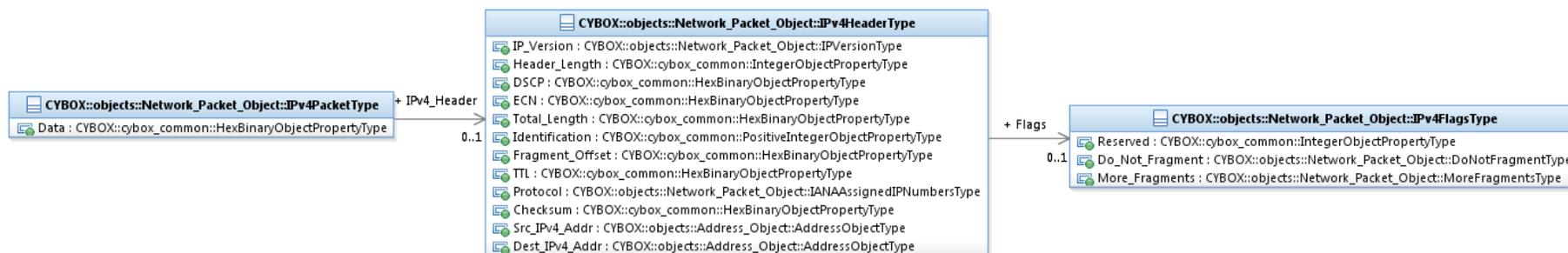


Figure 3-5. UML diagram for the `IPv4PacketType` class

The property table of the `IPv4PacketType` class is given in [Table 3-31](#).

Table 3-31. Properties of the `IPv4PacketType` class

Name	Type	Multiplicity	Description
IPv4_Header	<code>IPv4HeaderType</code>	0..1	The <code>IPv4_Header</code> property specifies the IPv4 header, which provides addressing, and internet modules use properties in the header to fragment and reassemble internet datagrams when necessary for transmission through small packet networks.

Data	cyboxCommon: HexBinaryObjectType	0..1	The <code>Data</code> property, which contains the data portion of an IP packet, is interpreted based on the value of the Protocol header property. Actual property values will probably be specified in the elements of the different network layers, but we provide a property here to capture any data as necessary.
-------------	-------------------------------------	------	---

3.3.2.1 IPv4HeaderType Class

The `IPv4HeaderType` class characterizes the IPv4 header, which provides addressing, and internet modules use fields in the header to fragment and reassemble internet datagrams when necessary for transmission through small packet networks. See <http://tools.ietf.org/html/rfc791> and <http://en.wikipedia.org/wiki/IPv4> for more information.

The property table of the `IPv4HeaderType` class is given in [Table 3-32](#).

Table 3-32. Properties of the `IPv4HeaderType` class

Name	Type	Multiplicity	Description
IP_Version	<code>IPVersionType</code>	0..1	The <code>IP_Version</code> property indicates the format of the internet header. For this class, the version is specified using the enumeration literal <code>IPv4(4)</code> .
Header_Length	cyboxCommon: <code>IntegerObjectType</code>	0..1	The <code>Header_Length</code> property specifies the length of IP packet header in 32 bit words. The minimum value is 5.
DSCP	cyboxCommon: <code>HexBinaryObjectType</code>	0..1	The <code>DSCP</code> property specifies the Differentiated Services Code Point (DSCP) as defined by http://tools.ietf.org/html/rfc2474 . New technologies are emerging that require real-time data streaming and therefore make use of the DSCP field. An example is Voice over IP (VoIP), which is used for interactive data voice exchange.
ECN	cyboxCommon: <code>HexBinaryObjectType</code>	0..1	The <code>ECN</code> property specifies the Explicit Congestion Notification as defined in http://tools.ietf.org/html/rfc3168 .

			The ECN allows end-to-end notification of network congestion without dropping packets. ECN is an optional feature that is only used when both endpoints support it and are willing to use it. It is only effective when supported by the underlying network.
Total_Length	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>Total_Length</code> property is a 16-bit property which specifies the entire datagram size, including header and data, in bytes.
Identification	cyboxCommon: PositiveIntegerObjectPropertyType	0..1	The <code>Identification</code> property is used to uniquely identify fragments of an original IP datagram.
Flags	IPv4FlagsType	0..1	The <code>Flags</code> property is used to specify the three-bit property used to control or identify fragments.
Fragment_Offset	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>Fragment_Offset</code> property is 13 bits long and specifies the offset of a particular fragment relative to the beginning of the original unfragmented IP datagram.
TTL	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>TTL</code> property that specifies an 8-bit property that helps prevent datagrams from persisting on an internet (it limits a datagram's lifetime).
Protocol	IANAAssignedIPNumbersType	0..1	The <code>Protocol</code> property specifies the protocol used in the data portion of the IP datagram. The type of this property is an enumerated list of IP protocol numbers as maintained by the Internet Assigned Numbers Authority (IANA).
Checksum	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>Checksum</code> property specifies a 16-bit checksum used for error-checking of the header.
Src_IPv4_Addr	AddressObj;AddressObjectType	0..1	The <code>Src_IPv4_Addr</code> property specifies the IPv4 address of

			the sender of the packet.
Dest_IPv4_Addr	AddressObj:AddressObjectType	0..1	The Desc_IPv4_Addr property specifies the IPv4 address of the receiver of the packet.
Option	IPv4OptionType	0..*	The Option property is variable in length with zero or more options. It is not often used.

3.3.2.2 IANAAssignedIPNumbersType Data Type

The IANAAssignedIPNumbersType data type specifies the Internet Protocol numbers. Its core value SHOULD be a literal found in the IANAAssignedIPNumbersTypeEnum enumeration. Its base type is the BaseObjectPropertyType data type, in order to permit complex (i.e. regular-expression based) specifications.

3.3.2.3 IPv4FlagsType Class

The IPv4FlagsType class specifies the flags that control or identify fragments in an IP packet. It is a three-bit field, each of the three bits are defined by a property with an enumeration value that indicates the meaning of whether or not the bit is set.

The property table of the IPv4FlagsType class is given in [Table 3-33](#).

Table 3-33. Properties of the IPv4FlagsType class

Name	Type	Multiplicity	Description
Reserved	cyboxCommon: IntegerObjectPropertyType	0..1	The Reserved property corresponds to bit 0: This bit value (0) is reserved and must be zero.
Do_Not_Fragment	DoNotFragmentType	0..1	The Do_Not_Fragment property corresponds to bit 1: This is the "don't fragment" bit.
More_Fragments	MoreFragmentsType	0..1	The More_Fragments property corresponds to bit 2: This is the "more fragments" bit.

3.3.2.4 DoNotFragmentType Data Type

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

3.3.2.5 MoreFragmentsType Data Type

The `MoreFragmentsType` data type specifies whether there are more fragments. Its core value SHOULD be a literal found in the `MoreFragmentsTypeEnum` enumeration. Its base type is the `BaseObjectPropertyType` data type, in order to permit complex (i.e. regular-expression based) specifications.

3.3.2.6 IPv4OptionType Class

The `IPv4OptionType` class specifies the zero or more options of the IPv4 packet.

The property table of the `IPv4OptionType` class is given in [Table 3-34](#).

Table 3-34. Properties of the `IPv4OptionType` class

Name	Type	Multiplicity	Description
Copy_Flag	<code>IPv4CopyFlagType</code>	0..1	The <code>Copy_Flag</code> property specifies the 1 bit which indicates that this option is copied into all fragments on fragmentation.
Class	<code>IPv4ClassType</code>	0..1	The <code>Class</code> property specifies the class type of the packet and corresponds to the 2 bit field, where 0 = control; 1 = reserved for future use; 2 = debugging and measurement; 3 = reserved for future use.
Option	<code>IPv4OptionsType</code>	0..1	The <code>Option</code> property specifies the optional header properties identified by an option type.

3.3.2.7 IPv4CopyFlagType Data Type

The `IPv4CopyFlagType` data type specifies the value of IPv4 copy flag. Its core value SHOULD be a literal found in the `IPv4CopyFlagTypeEnum` enumeration. Its base type is the `BaseObjectPropertyType` data type, in order to permit complex (i.e. regular-expression based) specifications.

3.3.2.8 IPv4ClassType Data Type

The `IPv4ClassType` data type specifies the IPv4 class. Its core value SHOULD be a literal found in the `IPv4ClassTypeEnum` enumeration. Its base type is the `BaseObjectPropertyType` data type, in order to permit complex (i.e. regular-expression based) specifications.

3.3.2.9 IPv4OptionsType Data Type

The `IPv4OptionsType` data type specifies the IPv4 options. Its core value SHOULD be a literal found in the `IPv4OptionsTypeEnum` enumeration. Its base type is the `BaseObjectPropertyType` data type, in order to permit complex (i.e. regular-expression based) specifications.

3.3.3 IPv6PacketType Class

The `IPv6PacketType` class is used to characterize a packet using the Internet Protocol version 6 (IPv6) which is intended to succeed IPv4. Like IPv4, it is a connectionless protocol for use on packet-switched link layer networks. See <http://tools.ietf.org/html/rfc3513>, <http://tools.ietf.org/html/rfc2460> and <http://en.wikipedia.org/wiki/IPv6> for more information. The UML diagram corresponding to the `IPv6PacketType` class is shown in **Figure 3-6**.

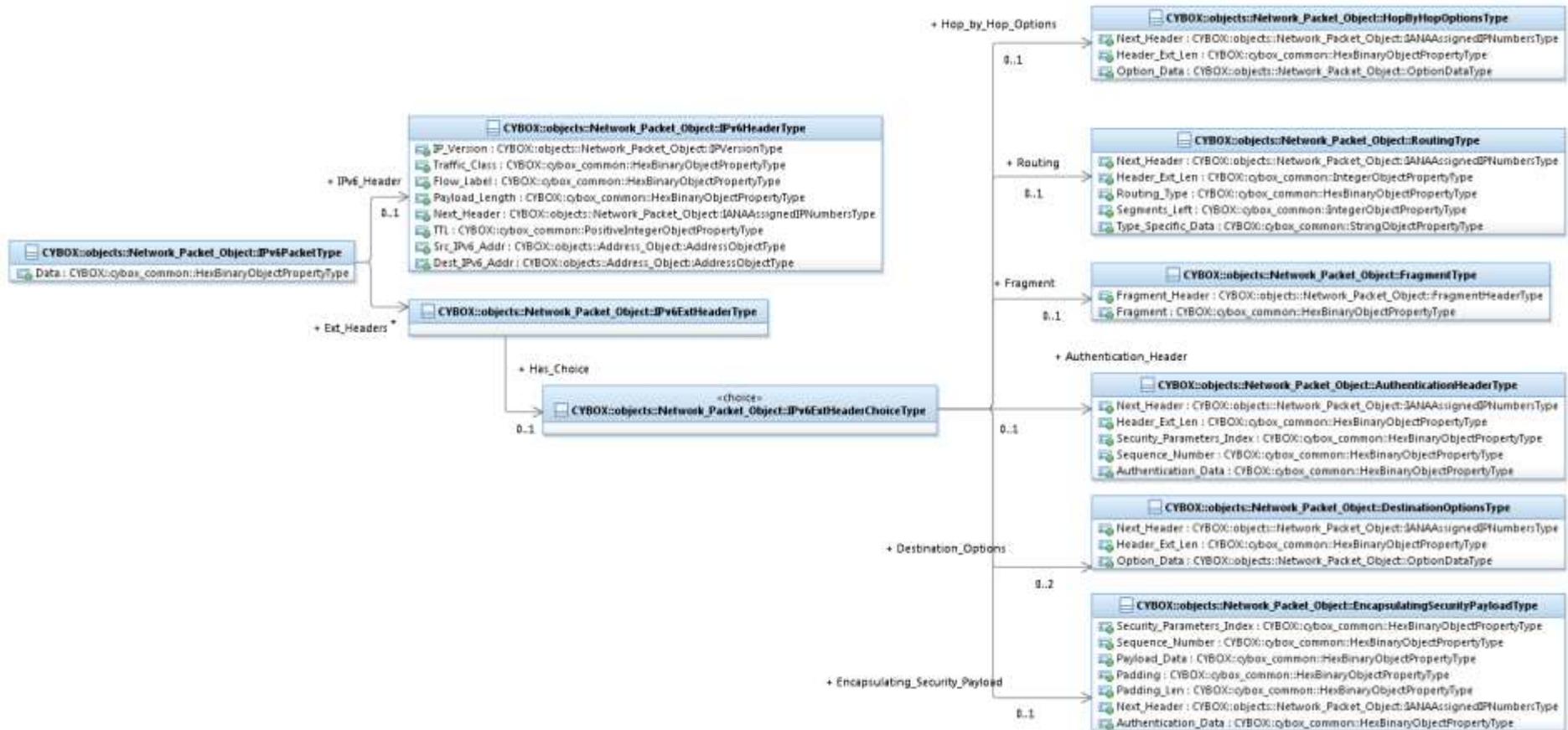


Figure 3-6. UML diagram for the IPv6PacketType class

The property table of the IPv6PacketType class is given in Table 3-35.

Table 3-35. Properties of the IPv6PacketType class

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

IPv6_Header	IPv6HeaderType	0..1	The <code>IPv6_Header</code> property specifies the IPv6 header, which is a simplification of the IPv4 header.
Ext_Headers	IPv6ExtHeaderType	0..*	The <code>Ext_Headers</code> property specifies the optional internet-layer information which is encoded in separate headers that may be placed between the IPv6 header and the upper-layer header in a packet.
Data	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>Data</code> property contains the data portion of an IP packet. Actual property values will probably be specified in the elements of the different network layers, but we provide a field here to capture any data as necessary.

3.3.3.1 IPv6HeaderType Class

The `IPv6HeaderType` class specifies the IPv6 header, and is a simplification of the IPv4 header.

The property table of the `IPv6HeaderType` class is given in [Table 3-36](#).

Table 3-36. Properties of the `IPv6HeaderType` class

Name	Type	Multiplicity	Description
IP_Version	IPVersionType	0..1	The <code>IP_Version</code> property specifies the 4-bit Internet Protocol version number. For this class, the version is always specified using the enumeration literal <code>IPv6 (6)</code> .
Traffic_Class	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>Traffic_Class</code> property specifies the 8-bit traffic class. Available for use by originating nodes and/or forwarding routers to identify and distinguish between different classes or priorities of IPv6 packets. See http://tools.ietf.org/html/rfc2460#section-7 for more information.
Flow_Label	cyboxCommon:	0..1	The <code>Flow_Label</code> property specifies the 20-bit flow label.

	HexBinaryObjectPropertyType		Used by a source to label sequences of packets for which it requests special handling by the IPv6 routers, such as non-default quality of service. See http://tools.ietf.org/html/rfc2460#section-6 for more information.
Payload_Length	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>Payload_Length</code> property specifies the length of the IPv6 payload (the rest of the packet following the IPv6 header) in octets as a 16-bit unsigned integer. Any extension headers are considered part of the payload.
Next_Header	IANAAssignedIPNumbersType	0..1	The <code>Next_Header</code> property specifies the type of header immediately following the IPv6 header. Uses the same values as the IPv4 protocol property.
TTL	cyboxCommon: PositiveIntegerObjectPropertyType	0..1	The <code>TTL</code> property (TTL/hop limit) specifies how many times a packet can be forwarded, as an 8-bit unsigned integer.
Src_IPv6_Addr	AddressObj:AddressObjectType	0..1	The <code>Src_IPv6_Addr</code> property specifies the 128-bit address of the originator of the packet.
Dest_IPv6_Addr	AddressObj:AddressObjectType	0..1	The <code>Dest_IPv6_Addr</code> property specifies the 128-bit address of the intended recipient of the packet.

3.3.3.2 IPv6ExtHeaderType Class

The `IPv6ExtHeaderType` class characterizes the optional internet-layer (IPv6) information that is encoded in separate headers. It is placed between the IPv6 header and the upper-layer header in a packet. An IPv6 packet may carry zero, one, or more extension headers. Each extension header is specified in a separate property as described in the table below. See <http://tools.ietf.org/html/rfc2460> for more information.

Name	Type	Multiplicity	Description
Has_Choice	IPv6ExtHeaderChoiceType	1	The <code>Has_Choice</code> property is associated with the class

			<p>IPv6ExtHeaderType. It indicates that there is a choice among the Hop_by_Hop_Options, Routing, Fragment, Destination_Options, Authentication_Header, and Encapsulating_Security_Payload properties.</p> <p>Only one of the properties of IPv6ExtHeaderChoiceType class can be populated at any time. See Section 1.2.3 for more detail.</p>
--	--	--	---

The IPv6ExtHeaderChoiceType class is the type of the Has_Choice property. In the UML model, this class is associated with the <<choice>> UML stereotype, which specifies that only one of the available properties of the IPv6ExtHeaderChoiceType class can be populated at any time. The property table of the IPv6ExtHeaderChoiceType class is given in Table 3-37.

Table 3-37. Properties of the IPv6ExtHeaderChoiceType class

Name	Type	Multiplicity	Description
Hop_by_Hop_Options	HopByHopOptionsType	0..1	<p>The Hop_by_Hop_Options property specifies the header which is used to carry optional information that must be examined by every node along a packet's delivery path. It carries a variable number of type-length-value (TLV) encoded options.</p> <p>Only one of the properties of IPv6ExtHeaderChoiceType can be populated.</p>
Routing	RoutingType	0..1	<p>The Routing property specifies the header which is used by an IPv6 source to list one or more intermediate nodes to be "visited" on the way to a packet's destination.</p> <p>Only one of the properties of IPv6ExtHeaderChoiceType can be populated.</p>

Fragment	FragmentType	0..1	<p>The <code>Fragment</code> property specifies the header which is used by an IPv6 source to send a packet larger than would fit in the path MTU. A fragment packet begins with an unfragmentable part consisting of the IPv6 header plus all extension headers up to and including the routing header. We don't include it for this property because the data is already stored in other properties. We provide the properties necessary for the Fragmentable Part.</p> <p>Only one of the properties of <code>IPv6ExtHeaderChoiceType</code> can be populated.</p>
Destination_Options	DestinationOptionsType	0..2	<p>The <code>Destination_Options</code> property specifies the header which is used to carry optional information that needs to be examined only by a packet's destination node(s).</p> <p>Only one of the properties of <code>IPv6ExtHeaderChoiceType</code> can be populated.</p>
Authentication_Header	AuthenticationHeaderType	0..1	<p>The <code>Authentication_Header</code> property specifies the header which is used for connectionless integrity and data origin authentication for IP datagrams and the protection against replays. See http://tools.ietf.org/html/rfc2402 for more information.</p> <p>Only one of the properties of <code>IPv6ExtHeaderChoiceType</code> can be populated.</p>
Encapsulating_Security_Payload	EncapsulatingSecurityPayloadType	0..1	<p>The <code>Encapsulating_Security_Payload</code> property (ESP) specifies the header which is used for confidentiality, data origin authentication, connectionless integrity, anti-replay service (a form of</p>

			<p>partial sequence integrity), and limited traffic flow confidentiality. See http://tools.ietf.org/html/rfc2406 for more information.</p> <p>Only one of the properties of <code>IPv6ExtHeaderChoiceType</code> can be populated.</p>
--	--	--	---

3.3.3.3 HopByHopOptionsType Class

The `HopByHopOptionsType` class characterizes the IPv6 Hop-by-Hop Options header which is used to carry optional information that must be examined by every node along a packet's delivery path.

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

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

Name	Type	Multiplicity	Description
Next_Header	<code>IANAAssignedIPNumbersType</code>	0..1	The <code>Next_Header</code> property specifies the type of header immediately following the Hop-by-Hop Options header. Uses the same values as the IPv4 Protocol property.
Header_Ext_Len	<code>cyboxCommon:HexBinaryObjectPropertyType</code>	0..1	The <code>Header_Ext_Len</code> property specifies the length of the Hop-by-Hop Options header in 8-octet units, not including the first 8 octets.
Option_Data	<code>OptionDataType</code>	0..*	The <code>Option_Data</code> property specifies a variable-length property, of length such that the complete Hop-by-Hop Options header is an integer multiple of 8 octets long. Contains one or more type-length-value (TLV)-encoded options.

3.3.3.3.1 OptionDataType Class

The `OptionDataType` class characterizes the variable-length fields associated with IPv6 extension headers (the Hop-by-Hop Options header and the Destination Options header). Contains one or more class-length-value (TLV)-encoded options.

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

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

Name	Type	Multiplicity	Description
Option_Type	<code>IPv6OptionType</code>	0..1	The <code>Option_Type</code> property specifies the type of option. This 8-bit Option Type identifier is internally encoded such that different bits have different meanings.
Option_Data_Len	<code>cyboxCommon:HexBinaryObjectPropertyType</code>	0..1	The <code>Option_Data_Len</code> property specifies the length of the Option Data property of this option, in octets.
Pad1	<code>Pad1Type</code>	0..1	The <code>Pad1</code> property specifies the option which is used to insert one octet of padding into the Options area of a header. The <code>Pad1</code> option does not have length and value properties.
PadN	<code>PadNType</code>	0..1	The <code>PadN</code> property specifies the option which is used to insert two or more octets of paddings into the Options area of a header.

3.3.3.3.2 Pad1Type Class

The `Pad1Type` class specifies whether one octet of padding is inserted into the Options area of a header. The `Pad1` option type does not have length and value fields.

The property table of the `Pad1Type` class is given in [Table 3-40](#).

Table 3-40. Properties of the `Pad1Type` class

Name	Type	Multiplicity	Description
Octet	<code>cyboxCommon:HexBinaryObjectPropertyType</code>	1	The <code>Octet</code> property specifies whether the <code>Pad1</code> option is used and also serves as the single octet of padding.

3.3.3.3.3 PadNType Class

The `PadNType` class specifies whether two or more octets of padding are inserted into the Options area of a header.

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

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

Name	Type	Multiplicity	Description
Octet	<code>cyboxCommon:</code> <code>HexBinaryObjectPropertyType</code>	0..1	The <code>Octet</code> property specifies the PadN option.
Option_Data_Length	<code>cyboxCommon:</code> <code>IntegerObjectPropertyType</code>	0..1	The <code>Option_Data_Length</code> property specifies the length of the padding. For N octets of padding, the <code>Option_Data_Length</code> property contains the value N-2.
Option_Data	<code>cyboxCommon:</code> <code>IntegerObjectPropertyType</code>	0..1	The <code>Option_Data</code> property specifies the actual padding. It consists of N-2 zero-valued octets.

3.3.3.4 RoutingType Class

The `RoutingType` class specifies the properties of the Routing header, which is used by an IPv6 source to list one or more intermediate nodes to be "visited" on the way to a packet's destination. See <http://tools.ietf.org/html/rfc2460> for more information.

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

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

Name	Type	Multiplicity	Description
Next_Header	<code>IANAAssignedIPNumbersType</code>	0..1	The <code>Next_Header</code> property specifies the type of header immediately following the Routing header. Uses the same values as the IPv4 Protocol property.

Header_Ext_Len	cyboxCommon: IntegerObjectPropertyType	0..1	The <code>Header_Ext_Len</code> property specifies the length of the Routing header in 8-octet units, not including the first 8 octets.
Routing_Type	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>Routing_Type</code> property specifies the 8-bit identifiers of a particular Routing header variant. Further definition will be added as required.
Segments_Left	cyboxCommon: IntegerObjectPropertyType	0..1	The <code>Segments_Left</code> property specifies the number of route segments remaining, i.e., number of explicitly listed intermediate nodes still to be visited before reaching the final destination.
Type_Specific_Data	cyboxCommon: StringObjectPropertyType	0..1	The <code>Type_Specific_Data</code> property is a variable length property of format determined by the Routing Type.

3.3.3.5 FragmentType Class

The `FragmentType` class specifies the properties of the Fragment header, which is used by an IPv6 source to send a packet larger than would fit in the path MTU. See <http://tools.ietf.org/html/rfc2460> for more information.

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

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

Name	Type	Multiplicity	Description
Fragment_Header	FragmentHeaderType	0..1	The <code>Fragment_Header</code> property specifies, for each fragment, a header containing next header information, the offset of the fragment, an M flag specifying whether or not it is the last fragment, and an identification value.
Fragment	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>Fragment</code> property specifies the fragment of the packet that corresponds to the fragment header. The length of the fragment must fit with the MTU of the path to the packets' destination.

3.3.3.5.1 FragmentHeaderType Class

The `FragmentHeaderType` class characterizes each fragment with a header containing next header information, the offset of the fragment, an M flag specifying whether or not it is the last fragment, and an identification value.

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

Table 3-44. Properties of the `FragmentHeaderType` class

Name	Type	Multiplicity	Description
Next_Header	<code>cyboxCommon:</code> <code>HexBinaryObjectPropertyType</code>	0..1	The <code>Next_Header</code> property specifies the type of header immediately following the Fragment header. Uses the same values as the IPv4 Protocol property.
Fragment_Offset	<code>cyboxCommon:</code> <code>HexBinaryObjectPropertyType</code>	0..1	The <code>Fragment_Offset</code> property specifies a 13-bit unsigned integer which is the offset, in 8-octet units, of the data following this header relative to the start of the Fragmentable Part or the original packet.
M_Flag	<code>MFlagType</code>	0..1	The <code>M_Flag</code> property specifies whether this is the last fragment or whether there are more fragments.
Identification	<code>cyboxCommon:</code> <code>HexBinaryObjectPropertyType</code>	0..1	The <code>Identification</code> property specifies a 32-bit Identification value for the packet generated by the source node.

3.3.3.5.2 MFlagType Data Type

The `MFlagType` data type specifies whether there are more fragments. Its core value SHOULD be a literal found in the `MFlagTypeEnum` enumeration. Its base type is the `BaseObjectPropertyType` data type, in order to permit complex (i.e. regular-expression based) specifications.

3.3.3.6 DestinationOptionsType Class

The `DestinationOptionsType` class specifies the properties for the IPv6 Destination Options header which is used to carry optional information that needs to be examined only by a packet's destination node(s).

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

Table 3-45. Properties of the `DestinationOptionsType` class

Name	Type	Multiplicity	Description
Next_Header	<code>IANAAssignedIPNumbersType</code>	0..1	The <code>Next_Header</code> property specifies the type of header immediately following the <code>Destination_Options</code> options header. Uses the same values as the IPv4 Protocol property.
Header_Ext_Len	<code>cyboxCommon:HexBinaryObjectPropertyType</code>	0..1	The <code>Header_Ext_Len</code> property specifies the length of the <code>Destination_Options</code> header in 8-octet units, not including the first 8 octets.
Option_Data	<code>OptionDataType</code>	0..*	The <code>Option_Data</code> property specifies a variable-length property of length such that the complete <code>Destinations_Options</code> header is an integer multiple of 8 octets long. Contains one or more type-length-value (TLV)-encoded options.

3.3.3.7 AuthenticationHeaderType Class

The `AuthenticationHeaderType` class is used to provide connectionless integrity and data origin authentication for IP datagrams and to provide protection against replays. See <http://tools.ietf.org/html/rfc2402> for more information.

The property table of the `AuthenticationHeaderType` class is given in [Table 3-46](#).

Table 3-46. Properties of the `AuthenticationHeaderType` class

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

Next_Header	IANAAssignedIPNumbersType	0..1	The <code>Next_Header</code> property specifies the type of header immediately following the Authentication header. Uses the same values as the IPv4 Protocol property.
Header_Ext_Len	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>Header_Ext_Len</code> property is an 8-bit property specifying the length of the AH in 32-bit words.
Security_Parameters_Index	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>Security_Parameters_Index</code> (SPI) property is an arbitrary 32-bit value that, in combination with the destination IP address and security protocol (AH), uniquely identifies the Security Association for this datagram. The set of SPI values in the range 1 through 255 are reserved by the Internet Assigned Numbers Authority (IANA) for future use.
Sequence_Number	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>Sequence_Number</code> property specifies the unsigned 32-bit property containing a monotonically increasing counter value (sequence number).
Authentication_Data	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>Authentication_Data</code> property specifies a variable-length property that contains the Integrity Check Value (ICV) for this packet. The field must be an integer multiple of 32 bits in length.

3.3.3.8 EncapsulatingSecurityPayloadType Class

The `EncapsulatingSecurityPayloadType` (ESP) class is used to provide confidentiality, data origin authentication, connectionless integrity, an anti-replay service (a form of partial sequence integrity), and limited traffic flow confidentiality. See <http://tools.ietf.org/html/rfc2406> for more information.

The property table of the `EncapsulatingSecurityPayloadType` class is given in [Table 3-47](#).

Table 3-47. Properties of the `EncapsulatingSecurityPayloadType` class

Name	Type	Multiplicity	Description
Security_Parameters_Index	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>Security_Parameters_Index</code> (SPI) property is an arbitrary 32-bit value that, in combination with the destination IP address and security protocol (ESP), uniquely identifies the Security Association for this datagram.
Sequence_Number	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>Sequence_Number</code> property is an unsigned 32-bit integer that contains a monotonically increasing counter value (sequence number).
Payload_Data	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>Payload_Data</code> property specifies a variable-length property containing data described by the <code>Next_Header</code> property.
Padding	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>Padding</code> property specifies the contents that can be used for various reasons, such as to fill in the plaintext as required by an encryption algorithm or to conceal the actual length of the payload.
Padding_Len	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>Padding_Len</code> property specifies the number of pad bytes immediately preceding it. Range is 0-255, where a value of zero indicates that no padding bytes are present.
Next_Header	IANAAssignedIPNumbersType	0..1	The <code>Next_Header</code> property specifies the type data contained in the <code>Payload_Data</code> property. Uses the same values as the IPv4 Protocol field.
Authentication_Data	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>Authentication_Data</code> property is a variable-length property containing an Integrity Check Value (ICV) computed over the ESP packet minus the Authentication Data.

3.3.3.9 IPv6OptionType Class

The `IPv6OptionType` class characterizes the meaning of each bit of the 8-bit IPv6 Options.

The property table of the `IPv6OptionType` class is given in [Table 3-48](#).

Table 3-48. Properties of the `IPv6OptionType` class

Name	Type	Multiplicity	Description
Do_Not_Recogn_Action	<code>IPv6DoNotRecogActionType</code>	0..1	The <code>Do_Not_Recogn_Action</code> property specifies the action to be taken if the processing IPv6 nodes do not recognize the Option Type. This information is internally encoded in the Option Type identifier (highest-order two bits) such that their highest-order two bits specify the action that must be taken.
Packet_Change	<code>IPv6PacketChangeType</code>	0..1	The <code>Packet_Change</code> property specifies the third highest order bit of the Option Data and indicates whether or not the Option Data of that option can change en-route to the packet's final destination.
Option_Byte	<code>cyboxCommon:HexBinaryObjectPropertyType</code>	0..1	The <code>Option_Byte</code> property may be used to specify the actual Option Type byte, with no explicit meaning attached. Meaning/interpretation provided by the <code>Do_Not_Recogn_Action</code> and <code>Packet_Change</code> properties.

3.3.3.10 IPv6DoNotRecogActionType Data Type

The `IPv6DoNotRecogActionType` data type specifies possible actions when an option is not recognized. Its core value SHOULD be a literal found in the `IPv6DoNotRecogActionTypeEnum` enumeration. Its base type is the `BaseObjectPropertyType` data type, in order to permit complex (i.e. regular-expression based) specifications.

3.3.3.11 IPv6PacketChangeType Data Type

The `IPv6PacketChangeType` data type specifies whether a packet has changed. Its core value SHOULD be a literal found in the `IPv6PacketChangeTypeEnum` enumeration. Its base type is the `BaseObjectPropertyType` data type, in order to permit complex (i.e. regular-expression based) specifications.

3.3.4 ICMPv4PacketType Class

The `ICMPv4PacketType` class characterizes the ICMP (v4), which is used to send error messages (e.g., a datagram cannot reach its destination), informational messages (e.g., timestamp information), or a traceroute message. See <http://www.networksorcery.com/enp/protocol/icmp.htm> for more information.

The UML diagram corresponding to the `ICMPv4PacketType` class is shown in **Figure 3-7**.

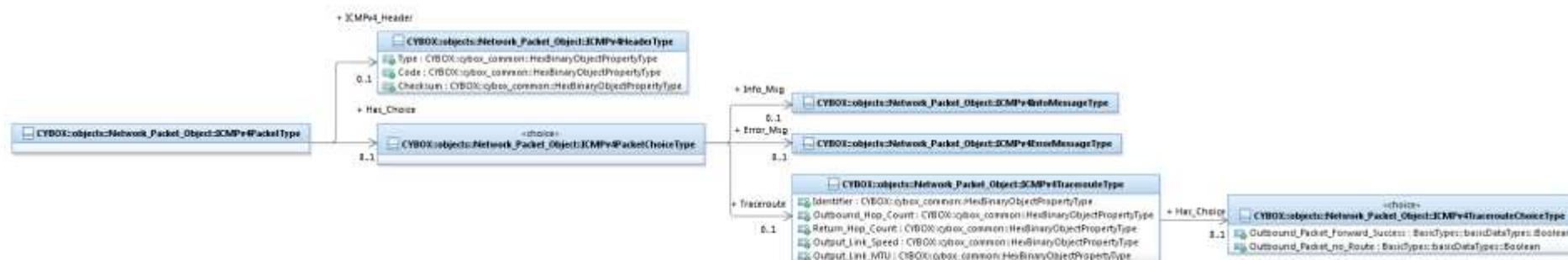


Figure 3-7. UML diagram for the `ICMPv4PacketType` class

The property table of the `ICMPv4PacketType` class is given in **Table 3-49**.

Table 3-49. Properties of the `ICMPv4PacketType` class

Name	Type	Multiplicity	Description
Has_Choice	<code>ICMPv4PacketChoiceType</code>	1	<p>The <code>Has_Choice</code> property is associated with the class <code>ICMPv4PacketChoiceType</code>. It indicates that there is a choice among the <code>Error_Msg</code>, <code>Info_Msg</code> and <code>Traceroute</code> properties.</p> <p>Only one of the properties of <code>ICMPv4PacketChoiceType</code> class can be populated at any time. See Section 1.2.3 for more detail.</p>

The `ICMPv4PacketChoiceType` class is the type of the `Has_Choice` property. In the UML model, this class is associated with the `<<choice>>` UML stereotype, which specifies that only one of the available properties of the `ICMPv4PacketChoiceType` class can be populated at any time. The property table of the `ICMPv4PacketChoiceType` class is given in [Table 3-50](#).

Table 3-50. Properties of the `ICMPv4PacketChoiceType` class

Name	Type	Multiplicity	Description
ICMPv4_Header	<code>ICMPv4HeaderType</code>	0..1	The <code>ICMPv4_Header</code> property specifies the type and code properties.
Error_Msg	<code>ICMPv4ErrorMessageType</code>	0..1	The <code>Error_Msg</code> property is used to specify an <code>ICMPv4ErrorMessageType</code> form of the <code>ICMPv4PacketType</code> class. Only one of the <code>Error_Msg</code> , <code>Info_Msg</code> and <code>Traceroute</code> properties can be populated.
Info_Msg	<code>ICMPv4InfoMessageType</code>	0..1	The <code>Info_Msg</code> property is used to specify an <code>ICMPv4InfoMessageType</code> form of the <code>ICMPv4PacketType</code> class. Only one of the <code>Error_Msg</code> , <code>Info_Msg</code> and <code>Traceroute</code> properties can be populated.
Traceroute	<code>ICMPv4TracerouteType</code>	0..1	The <code>Traceroute</code> property is used to specify an <code>ICMPv4TracerouteType</code> form of the <code>ICMPv4PacketType</code> class. Only one of the <code>Error_Msg</code> , <code>Info_Msg</code> and <code>Traceroute</code> properties can be populated.

3.3.4.1 ICMPv4HeaderType Class

The `ICMPv4HeaderType` class specify the actual ICMP header bytes corresponding to the ICMP class, ICMP code, and to the checksum.

The property table of the `ICMPv4HeaderType` class is given in [Table 3-51](#).

Table 3-51. Properties of the `ICMPv4HeaderType` class

Name	Type	Multiplicity	Description
Type	<code>cyboxCommon:HexBinaryObjectPropertyType</code>	0..1	The <code>Type</code> property specifies the format of the ICMP message.
Code	<code>cyboxCommon:HexBinaryObjectPropertyType</code>	0..1	The <code>Code</code> property specifies the code of the ICMP message.
Checksum	<code>cyboxCommon:HexBinaryObjectPropertyType</code>	0..1	The <code>Checksum</code> property specifies the checksum (16 bits) of the ICMP message.

3.3.4.2 ICMPv4ErrorMessageType Class

The `ICMPv4ErrorMessageType` class specifies ICMP error messages, which include destination unreachable messages, source quench messages, redirect messages, and time exceeded messages. The UML diagram corresponding to the `ICMPv4ErrorMessageType` class is shown in [Figure 3-8](#).

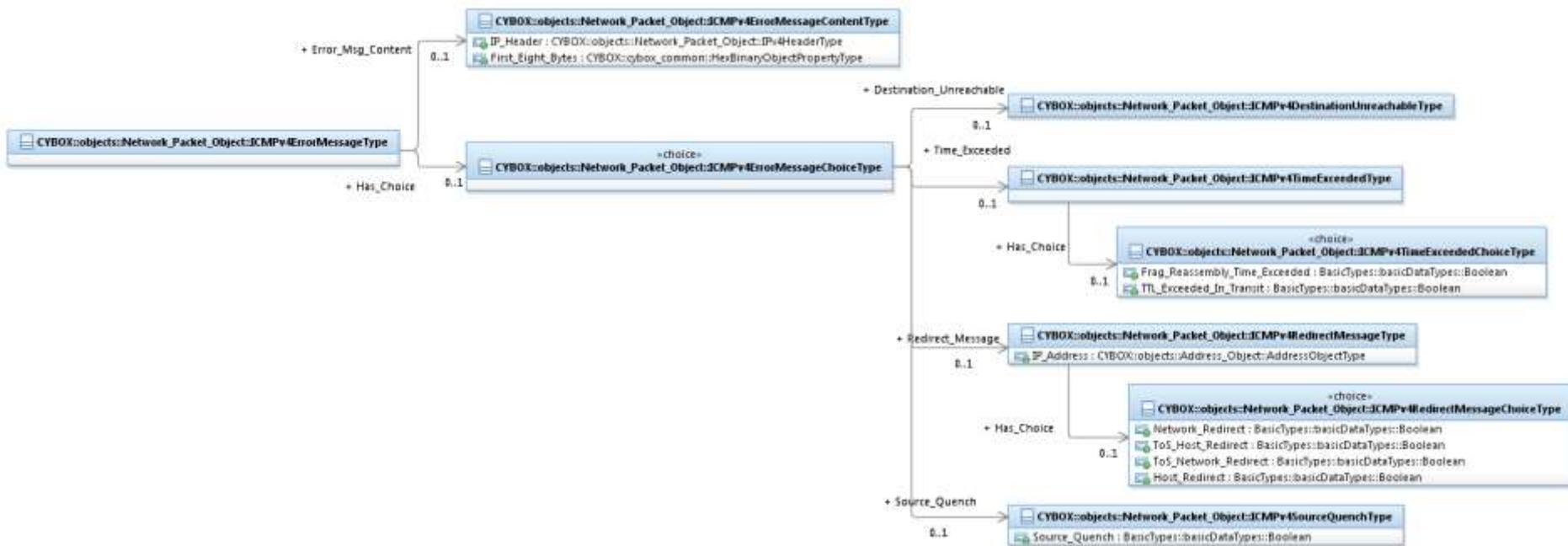


Figure 3-8. UML diagram of the *ICMPv4ErrorMessageType* class

The property table of the *ICMPv4ErrorMessageType* class is given in [Table 3-52](#).

Table 3-52. Properties of the *ICMPv4ErrorMessageType* class

Name	Type	Multiplicity	Description
Error_Msg_Content	ICMPv4ErrorMessageContentType	0..1	The <i>Error_Msg_Content</i> property specifies content common to all ICMP error message types are defined here. Properties that are specific to individual message types are defined separately under each message type.
Has_Choice	ICMPv4ErrorMessageChoiceType	0..1	The <i>Has_Choice</i> property is associated with the class <i>ICMPv4ErrorMessageChoiceType</i> . It indicates that there is

			<p>a choice among <code>Destination_Unreachable</code>, <code>Source_Quench</code>, <code>Redirect_Message</code> and <code>Time_Exceeded</code> properties.</p> <p>Only one of the properties of <code>ICMPv4ErrorMessageChoiceType</code> class can be populated at any time. See Section 1.2.3 for more detail.</p>
--	--	--	--

The `ICMPv4ErrorMessageChoiceType` class is the type of the `Has_Choice` property. In the UML model, this class is associated with the <<choice>> UML stereotype, which specifies that only one of the available properties of the `ICMPv4ErrorMessageChoiceType` class can be populated at any time. The property table of the `ICMPv4ErrorMessageChoiceType` class is given in Table 3-53.

Table 3-53. Properties of the `ICMPv4ErrorMessageChoiceType` class

Name	Type	Multiplicity	Description
Destination_Unreachable	<code>ICMPv4DestinationUnreachableType</code>	0..1	<p>The <code>Destination_Unreachable</code> property specifies a destination unreachable message, which is generated by the host or its inbound gateway to inform the client that the destination is unreachable for some reason. See http://en.wikipedia.org/wiki/ICMP_Destination_Unreachable for more information.</p> <p>Only one of the <code>Destination_Unreachable</code>, <code>Source_Quench</code>, <code>Redirect_Message</code> and <code>Time_Exceeded</code> properties can be populated.</p>
Source_Quench	<code>ICMPv4SourceQuenchType</code>	0..1	<p>The <code>Source_Quench</code> property specifies a source quench message that requests the sender decrease the rate of messages sent to a router or host. This message may be generated if a router or host does not have sufficient buffer space to process the request or may occur if the router or host buffer is approaching</p>

			<p>its limit. See http://en.wikipedia.org/wiki/ICMP_Source_Quench for more information.</p> <p>Only one of the <code>Destination_Unreachable</code>, <code>Source_Quench</code>, <code>Redirect_Message</code> and <code>Time_Exceeded</code> properties can be populated.</p>
Redirect_Message	<code>ICMPv4RedirectMessageType</code>	0..1	<p>The <code>Redirect_Message</code> property specifies a redirect message which is used to send data packets on an alternative route. This ICMP redirect message informs a host to update its routing information.</p> <p>Only one of the <code>Destination_Unreachable</code>, <code>Source_Quench</code>, <code>Redirect_Message</code> and <code>Time_Exceeded</code> properties can be populated.</p>
Time_Exceeded	<code>ICMPv4TimeExceededType</code>	0..1	<p>The <code>Time_Exceeded</code> property specifies a time exceeded message, which is generated by a gateway to inform the source of a datagram that the datagram has been discarded due to the time to live property reaching zero. A time exceeded message may also be sent by a host if it fails to reassemble a fragmented datagram within its time limit. See http://en.wikipedia.org/wiki/ICMP_Time_Exceeded for more information.</p> <p>Only one of the <code>Destination_Unreachable</code>, <code>Source_Quench</code>, <code>Redirect_Message</code> and <code>Time_Exceeded</code> properties can be populated.</p>

3.3.4.3 ICMPv4ErrorMessageContentType Class

The `ICMPv4ErrorMessageContentType` class specifies properties common to all types of ICMPv4 error messages.

The property table of the `ICMPv4ErrorMessageContentType` class is given in [Table 3-54](#).

Table 3-54. Properties of the `ICMPv4ErrorMessageContentType` class

Name	Type	Multiplicity	Description
IP_Header	<code>IPv4HeaderType</code>	0..1	The <code>IP_Header</code> property specifies the IP header from the original datagram.
First_Eight_Bytes	<code>cyboxCommon:HexBinaryObjectPropertyType</code>	0..1	The <code>First_Eight_Bytes</code> property specifies the first 8 bytes of the original datagram's data.

3.3.4.3.1 ICMPv4DestinationUnreachableType Class

The `ICMPv4DestinationUnreachableType` class specifies the Destination Unreachable error message; ICMP class=3. The UML diagram corresponding to the `ICMPv4DestinationUnreachableType` class is shown in [Figure 3-9](#).

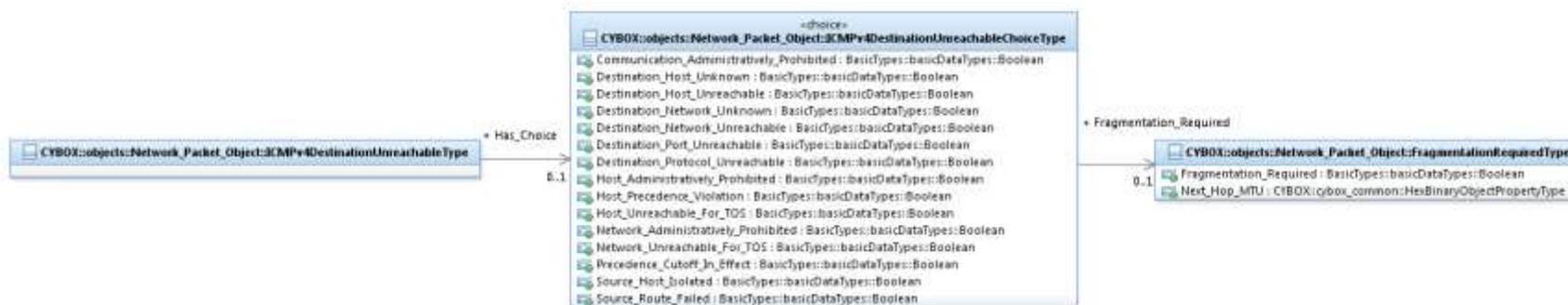


Figure 3-9. UML diagram for `ICMPv4DestinationUnreachableType` class

The property table of the `ICMPv4DestinationUnreachableType` class is given in [Table 3-55](#).

Table 3-55. Properties of the `ICMPv4DestinationUnreachableType` class

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

Has_Choice	ICMPv4DestinationUnreachableChoiceType	0..1	<p>The <code>Has_Choice</code> property is associated with the class <code>ICMPv4DestinationUnreachableChoiceType</code>. It indicates that there is a choice among the properties of <code>ICMPv4DestinationUnreachableChoiceType</code>.</p> <p>Only one of the properties of <code>ICMPv4DestinationUnreachableChoiceType</code> class can be populated at any time. See Section 1.2.3 for more detail.</p>
-------------------	--	------	--

The property table of the `ICMPv4DestinationUnreachableChoiceType` class is given in [Table 3-56](#).

Table 3-56. Properties of the `ICMPv4DestinationUnreachableChoiceType` class

Name	Type	Multiplicity	Description
Destination_Network_Unreachable	<code>basicDataTypes:Boolean</code>	0..1	<p>The <code>Destination_Network_Unreachable</code> property specifies whether the subtype of the message is “destination network unreachable” (code=0).</p> <p>Only one of the properties of <code>ICMPv4DestinationUnreachableChoiceType</code> class can be populated.</p>
Destination_Host_Unreachable	<code>basicDataTypes:Boolean</code>	0..1	<p>The <code>Destination_Host_Unreachable</code> property specifies whether the subtype of the message is “destination host unreachable” (code=1).</p> <p>Only one of the properties of <code>ICMPv4DestinationUnreachableChoiceType</code> class can be populated.</p>

Destination_Protocol_Unreachable	basicDataTypes:Boolean	0..1	<p>The Destination_Protocol_Unreachable property specifies whether the subtype of the message is “destination protocol unreachable” (code=2).</p> <p>Only one of the properties of ICMPv4DestinationUnreachableChoiceType class can be populated.</p>
Destination_Port_Unreachable	basicDataTypes:Boolean	0..1	<p>The Destination_Port_Unreachable property specifies whether the subtype of the message is “destination port unreachable” (code=3).</p> <p>Only one of the properties of ICMPv4DestinationUnreachableChoiceType class can be populated.</p>
Fragmentation_Required	FragmentationRequiredType	0..1	<p>The Fragmentation_Required property specifies whether the subtype of the message is “fragmentation required” (code=4). This property has an additional field (Next-Hop MTU), as well as a boolean value indicating this subtype.</p> <p>Only one of the properties of ICMPv4DestinationUnreachableChoiceType class can be populated.</p>
Source_Route_Failed	basicDataTypes:Boolean	0..1	<p>The Source_Route_Failed property specifies whether the subtype of the message is “source route failed” (code=5).</p> <p>Only one of the properties of ICMPv4DestinationUnreachableChoiceType class can be populated.</p>

Destination_Network_Unknown	basicDataTypes:Boolean	0..1	<p>The <code>Destination_Network_Unknown</code> property specifies whether the subtype of the message is “destination network unknown” (code=6).</p> <p>Only one of the properties of <code>ICMPv4DestinationUnreachableChoiceType</code> class can be populated.</p>
Destination_Host_Unknown	basicDataTypes:Boolean	0..1	<p>The <code>Destination_Host_Unknown</code> property specifies whether the subtype of the message is “destination host unknown” (code=7).</p> <p>Only one of the properties of <code>ICMPv4DestinationUnreachableChoiceType</code> class can be populated.</p>
Source_Host_Isolated	basicDataTypes:Boolean	0..1	<p>The <code>Source_Host_Isolated</code> property specifies whether the subtype of the message is “source host isolated” (code=8).</p> <p>Only one of the properties of <code>ICMPv4DestinationUnreachableChoiceType</code> class can be populated.</p>
Network_Administratively_Prohibited	basicDataTypes:Boolean	0..1	<p>The <code>Network_Administratively_Prohibited</code> property specifies whether the subtype of the message is “network administratively prohibited” (code=9).</p> <p>Only one of the properties of <code>ICMPv4DestinationUnreachableChoiceType</code> class can be populated.</p>

Host_Administratively_Prohibited	basicDataTypes: Boolean	0..1	<p>The <code>Host_Administratively_Prohibited</code> property specifies whether the subtype of the message is “host administratively prohibited” (code=10).</p> <p>Only one of the properties of <code>ICMPv4DestinationUnreachableChoiceType</code> class can be populated.</p>
Network_Unreachable_For_TOS	basicDataTypes: Boolean	0..1	<p>The <code>Network_Unreachable_For_TOS</code> property specifies whether the subtype of the message is “network unreachable for TOS” (code=11).</p> <p>Only one of the properties of <code>ICMPv4DestinationUnreachableChoiceType</code> class can be populated.</p>
Host_Unreachable_For_TOS	basicDataTypes: Boolean	0..1	<p>The <code>Host_Unreachable_For_TOS</code> specifies whether the subtype of the message is “host unreachable for TOS” (code=12).</p> <p>Only one of the properties of <code>ICMPv4DestinationUnreachableChoiceType</code> class can be populated.</p>
Communication_Administratively_Prohibited	basicDataTypes: Boolean	0..1	<p>The <code>Communication_Administratively_Prohibited</code> property specifies whether the subtype of the message is “communication administratively prohibited” (code=13).</p> <p>Only one of the properties of <code>ICMPv4DestinationUnreachableChoiceType</code> class can be populated.</p>

Host_Precedence_Violation	<code>basicDataTypes:Boolean</code>	0..1	<p>The <code>Host_Precedence_Violation</code> property specifies whether the subtype of the message is "host precedence violation" (code=14).</p> <p>Only one of the properties of <code>ICMPv4DestinationUnreachableChoiceType</code> class can be populated.</p>
Precedence_Cutoff_In_Effect	<code>basicDataTypes:Boolean</code>	0..1	<p>The <code>Precedence_Cutoff_In_Effect</code> property specifies whether the subtype of the message is "precedence cutoff in effect" (code=15).</p> <p>Only one of the properties of <code>ICMPv4DestinationUnreachableChoiceType</code> class can be populated.</p>

3.3.4.3.2 FragmentationRequiredType Class

The `FragmentationRequiredType` class further specifies an ICMP destination unreachable (class=3) message as the subtype fragmentation required message (code=4) by providing a Next-Hop MTU field.

The property table of the `FragmentationRequiredType` class is given in [Table 3-57](#).

Table 3-57. Properties of the `FragmentationRequiredType` class

Name	Type	Multiplicity	Description
Fragmentation_Required	<code>basicDataTypes:Boolean</code>	0..1	The <code>Fragmentation_Required</code> property specifies whether the subtype of the destination unreachable ICMP message is "fragmentation required".
Next_Hop_MTU	<code>cyboxCommon:HexBinaryObjectPropertyType</code>	0..1	The <code>Next_Hop_MTU</code> property contains the MTU of the next-hop network when a code 4 error (fragmentation required) occurs.

3.3.4.3.3 ICMPv4SourceQuenchType Class

The `ICMPv4SourceQuenchType` class specifies the Source Quench (congestion control) error message; ICMP class=4.

The property table of the `ICMPv4SourceQuenchType` class is given in [Table 3-58](#).

Table 3-58. Properties of the `ICMPv4SourceQuenchType` class

Name	Type	Multiplicity	Description
Source_Quench	<code>basicDataTypes:Boolean</code>	0..1	The <code>Source_Quench</code> property specifies whether the subtype of the error message is a source quench (code=0).

3.3.4.3.4 ICMPv4RedirectMessageType Class

The `ICMPv4RedirectMessageType` class specifies the Redirect Message error message; ICMP class=5. The UML diagram corresponding to the `ICMPv4RedirectMessageType` class is shown in [Figure 3-8](#).

The property table of the `ICMPv4RedirectMessageType` class is given in [Table 3-59](#).

Table 3-59. Properties of the `ICMPv4RedirectMessageType` class

Name	Type	Multiplicity	Description
IP_Address	<code>AddressObj :</code> <code>AddressObjectType</code>	0..1	The <code>IP_Address</code> property specifies the IP address is the 32-bit address of the gateway to which the redirection should be sent.
Has_Choice	<code>ICMPv4RedirectMessageChoiceType</code>	0..1	<p>The <code>Has_Choice</code> property is associated with the class <code>ICMPv4RedirectMessageChoiceType</code>. It indicates that there is a choice among the <code>Network_Redirect</code>, <code>Host_Redirect</code>, <code>ToS_Network_Redirect</code>, and <code>ToS_Host_Redirect</code> properties.</p> <p>Only one of the properties of <code>ICMPv4RedirectMessageChoiceType</code> class can be populated</p>

			at any time. See Section 1.2.3 for more detail.
--	--	--	---

The `ICMPv4RedirectMessageChoiceType` class is the type of the `Has_Choice` property. In the UML model, this class is associated with the `<<choice>>` UML stereotype, which specifies that only one of the available properties of the `ICMPv4RedirectMessageChoiceType` class can be populated at any time. The property table of the `ICMPv4RedirectMessageChoiceType` class is given in [Table 3-60](#).

Table 3-60. Properties of the `ICMPv4RedirectMessageChoiceType` class

Name	Type	Multiplicity	Description
Network_Redirect	basicDataTypes: Boolean	0..1	The <code>Network_Redirect</code> property specifies whether the subtype of the message is a redirect datagram for the network (code=0). Only one of the <code>Network_Redirect</code> , <code>Host_Redirect</code> , <code>ToS_Network_Redirect</code> and <code>ToS_Host_Redirect</code> properties can be populated.
Host_Redirect	basicDataTypes: Boolean	0..1	The <code>Host_Redirect</code> property specifies whether the subtype of the message is a redirect datagram for the host (code=1). Only one of the <code>Network_Redirect</code> , <code>Host_Redirect</code> , <code>ToS_Network_Redirect</code> and <code>ToS_Host_Redirect</code> properties can be populated.
ToS_Network_Redirect	basicDataTypes: Boolean	0..1	The <code>ToS_Network_Redirect</code> property specifies whether the subtype of the message is a redirect datagram for the TOS and network (code=2). Only one of the <code>Network_Redirect</code> , <code>Host_Redirect</code> , <code>ToS_Network_Redirect</code> and <code>ToS_Host_Redirect</code> properties can be populated.

ToS_Host_Redirect	basicDataTypes: Boolean	0..1	The <code>ToS_Host_Redirect</code> property specifies whether the subtype of the message is a redirect datagram for the TOS and host (code=3). Only one of the <code>Network_Redirect</code> , <code>Host_Redirect</code> , <code>ToS_Network_Redirect</code> and <code>ToS_Host_Redirect</code> properties can be populated.
--------------------------	----------------------------	------	--

3.3.4.3.5 ICMPv4TimeExceededType Class

The `ICMPv4TimeExceededType` class specifies the Time Exceeded error message; ICMP class=11.

In CybOX 2.1.1, the properties, `TTL_Exceeded_In_Transit` and `Frag_Reassembly_Time_Exceeded` are mutually exclusive, i.e., only one property can be populated. This restriction is based on that there are two different types of time exceeded messages. In future releases, it will probably be replaced with one property and a corresponding enumeration.

The property table of the `ICMPv4TimeExceededType` class is given in [Table 3-61](#).

Table 3-61. Properties of the `ICMPv4TimeExceededType` class

Name	Type	Multiplicity	Description
TTL_Exceeded_In_Transit	basicDataTypes: Boolean	0..1	The <code>TTL_Exceeded_In_Transit</code> property specifies that the time-to-live was exceeded in transit (code=0).
Frag_Reassembly_Time_Exceeded	basicDataTypes: Boolean	0..1	The <code>Frag_Reassembly_Time_Exceeded</code> property specifies that the fragment reassembly time was exceeded (code=1).

3.3.4.4 ICMPv4InfoMessageType Class

The `ICMPv4InfoMessageType` class specifies ICMP informational messages, which include echo request/reply, timestamp request/reply, and address mask request/reply. The UML diagram corresponding to the `ICMPv4InfoMessageType` class is shown in [Figure 3-10](#).

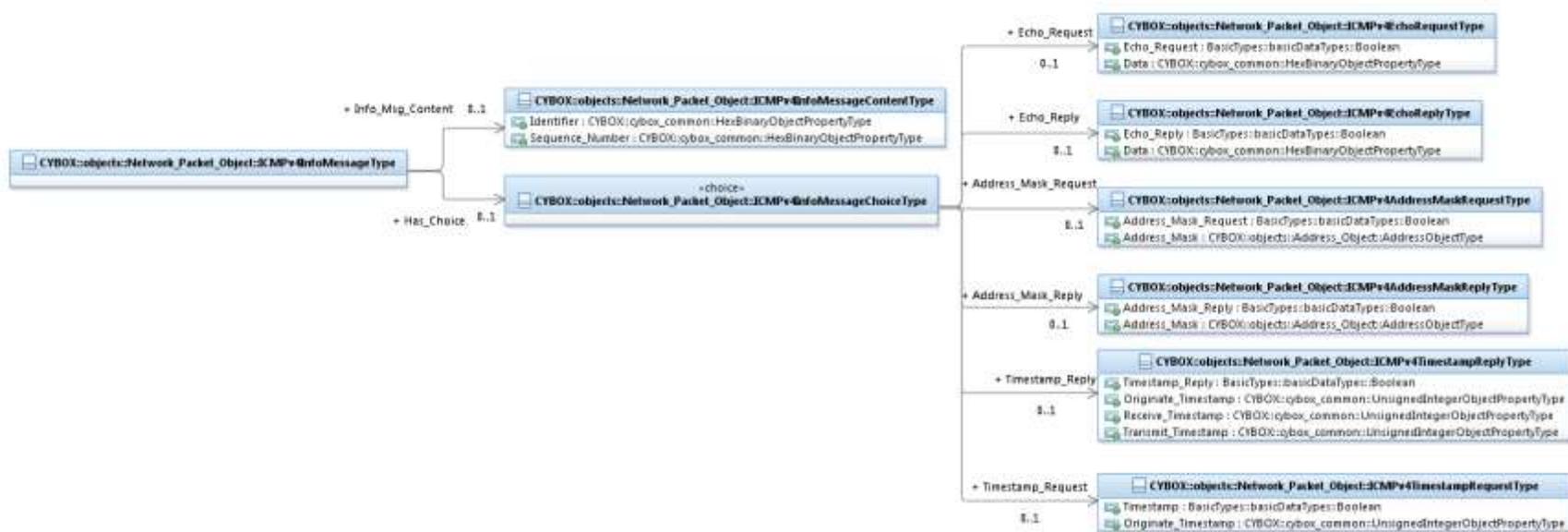


Figure 3-10. UML diagram of the *ICMPv4InfoMessageType* class

The property table of the *ICMPv4InfoMessageType* class is given in [Table 3-62](#).

Table 3-62. Properties of the *ICMPv4InfoMessageType* class

Name	Type	Multiplicity	Description
Info_Msg_Content	ICMPv4InfoMessageContentType	0..1	The <i>Info_Msg_Content</i> property specifies properties that are common to all ICMP informational. Properties that are specific to individual messages are defined separately under each message type.
Has_Choice	ICMPv4InfoMessageChoiceType	1	The <i>Has_Choice</i> property is associated with the class <i>ICMPv4InfoMessageChoiceType</i> . It indicates that there is a choice among the <i>Echo_Reply</i> , <i>Echo_Request</i> , <i>Timestamp_Request</i> , <i>Timestamp_Reply</i> , <i>Address_Mask_Request</i> and <i>Address_Mask_Reply</i>

			<p>properties .</p> <p>Only one of the properties of <code>ICMPv4InfoMessageChoiceType</code> class can be populated at any time. See Section 1.2.3 for more detail.</p>
--	--	--	--

The `ICMPv4InfoMessageChoiceType` class is the type of the `Has_Choice` property. In the UML model, this class is associated with the <<choice>> UML stereotype, which specifies that only one of the available properties of the `ICMPv4InfoMessageChoiceType` class can be populated at any time. The property table of the `ICMPv4InfoMessageChoiceType` class is given in Table 3-63.

Table 3-63. Properties of the `ICMPv4InfoMessageChoiceType` class

Name	Type	Multiplicity	Description
Info_Msg_Content	<code>ICMPv4InfoMessageContentType</code>	0..1	The <code>Info_Msg_Content</code> property specifies properties that are common to all ICMP informational. Properties that are specific to individual messages are defined separately under each message type.
Echo_Reply	<code>ICMPv4EchoReplyType</code>	0..1	<p>The <code>Echo_Reply</code> property specifies an echo reply message (type=0). These messages are also known as "ping".</p> <p>Only one of the <code>Echo_Reply</code>, <code>Echo_Request</code>, <code>Timestamp_Request</code>, <code>Timestamp_Reply</code>, <code>Address_Mask_Request</code> and <code>Address_Mask_Reply</code> properties can be populated.</p>
Echo_Request	<code>ICMPv4EchoRequestType</code>	0..1	<p>The <code>Echo_Request</code> property specifies an echo reply message (type=8). These messages are also known as "ping".</p> <p>Only one of the <code>Echo_Reply</code>, <code>Echo_Request</code>, <code>Timestamp_Request</code>, <code>Timestamp_Reply</code>, <code>Address_Mask_Request</code> and <code>Address_Mask_Reply</code> properties can be populated.</p>

Timestamp_Request	ICMPv4TimestampRequestType	0..1	<p>The <code>Timestamp_Request</code> property specifies a timestamp request used for time synchronization (type=13).</p> <p>Only one of the <code>Echo_Reply</code>, <code>Echo_Request</code>, <code>Timestamp_Request</code>, <code>Timestamp_Reply</code>, <code>Address_Mask_Request</code> and <code>Address_Mask_Reply</code> properties can be populated.</p>
Timestamp_Reply	ICMPv4TimestampReplyType	0..1	<p>The <code>Timestamp_Reply</code> property specifies a timestamp reply which replies to a timestamp request message (type=14).</p> <p>Only one of the <code>Echo_Reply</code>, <code>Echo_Request</code>, <code>Timestamp_Request</code>, <code>Timestamp_Reply</code>, <code>Address_Mask_Request</code> and <code>Address_Mask_Reply</code> properties can be populated.</p>
Address_Mask_Request	ICMPv4AddressMaskRequestType	0..1	<p>The <code>Address_Mask_Request</code> property specifies an address mask request which is a query message normally sent by a host to a router in order to obtain an appropriate subnet mask (type=17).</p> <p>Only one of the <code>Echo_Reply</code>, <code>Echo_Request</code>, <code>Timestamp_Request</code>, <code>Timestamp_Reply</code>, <code>Address_Mask_Request</code> and <code>Address_Mask_Reply</code> properties can be populated.</p>
Address_Mask_Reply	ICMPv4AddressMaskReplyType	0..1	<p>The <code>Address_Mask_Reply</code> property specifies an address mask reply which is used to reply to an address mask request message with an appropriate subnet mask (type=18).</p> <p>Only one of the <code>Echo_Reply</code>, <code>Echo_Request</code>, <code>Timestamp_Request</code>, <code>Timestamp_Reply</code>, <code>Address_Mask_Request</code> and <code>Address_Mask_Reply</code> properties can be populated.</p>

3.3.4.4.1 ICMPv4InfoMessageContentType Class

The `ICMPv4InfoMessageContentType` class specifies properties common to all types of ICMPv4 informational messages.

The property table of the `ICMPv4InfoMessageContentType` class is given in [Table 3-64](#).

Table 3-64. Properties of the `ICMPv4InfoMessageContentType` class

Name	Type	Multiplicity	Description
Identifier	<code>cyboxCommon:</code> <code>HexBinaryObjectPropertyType</code>	0..1	The <code>Identifier</code> property specifies a 16-bit identifier, which is combined with the sequence number, and called the "quench" for echo reply and echo request.
Sequence_Number	<code>cyboxCommon:</code> <code>HexBinaryObjectPropertyType</code>	0..1	The <code>Sequence_Number</code> property specifies a 16-bit sequence number. The identifier and sequence number can be used by the client to match the reply with the request that caused the reply.

3.3.4.4.2 ICMPv4EchoReplyType Class

the `ICMPv4EchoReplyType` class specifies the echo reply v4 informational message (used to ping); ICMP class=0.

The property table of the `ICMPv4EchoReplyType` class is given in [Table 3-65](#).

Table 3-65. Properties of the `ICMPv4EchoReplyType` class

Name	Type	Multiplicity	Description
Echo_Reply	<code>basicDataTypes:Boolean</code>	0..1	The <code>Echo_Reply</code> property specifies whether the subtype of the message is an echo reply message (code=0).
Data	<code>cyboxCommon:</code> <code>HexBinaryObjectPropertyType</code>	0..1	The <code>Data</code> property specifies the data of the different kind of answers given with an ICMP Echo Reply message. It can be of arbitrary length (but less than the MTU of the network).

3.3.4.4.3 ICMPv4EchoRequestType Class

The `ICMPv4EchoRequestType` class specifies the echo request informational message (used to ping); ICMP class=8.

The property table of the `ICMPv4EchoRequestType` class is given in [Table 3-66](#).

Table 3-66. Properties of the `ICMPv4EchoRequestType` class

Name	Type	Multiplicity	Description
Echo_Request	<code>basicDataTypes:Boolean</code>	0..1	The <code>Echo_Request</code> property specifies whether the subtype of the message is an echo request message (code=8).
Data	<code>cyboxCommon:HexBinaryObjectPropertyType</code>	0..1	The <code>Data</code> property specifies the data of the different kind of answers given with an ICMP Echo Request message. It can be of arbitrary length (but less than the MTU of the network).

3.3.4.4.4 ICMPv4TimestampRequestType Class

The `ICMPv4TimestampRequestType` class specifies the timestamp request informational message; ICMP class=13.

The property table of the `ICMPv4TimestampRequestType` class is given in [Table 3-67](#).

Table 3-67. Properties of the `ICMPv4TimestampRequestType` class

Name	Type	Multiplicity	Description
Timestamp	<code>basicDataTypes:Boolean</code>	0..1	The <code>Timestamp</code> property specifies whether the subtype of the message is a timestamp request message (code=13).
Originate_Timestamp	<code>cyboxCommon:</code>	0..1	The <code>Originate_Timestamp</code> property specifies number of milliseconds since midnight UT, in 32-bits.

	UnsignedIntegerObjectPropertyType		The originate timestamp is the time the sender last touched the message before sending it. If the time is not available in milliseconds or cannot be provided with respect to midnight UT, then any time can be inserted in a timestamp provided the high order bit of the timestamp is also set to indicate this non-standard value.
--	-----------------------------------	--	---

3.3.4.4.5 ICMPv4TimestampReplyType Class

The `ICMPv4TimestampReplyType` class specifies the timestamp reply informational message; ICMP class=14.

The property table of the `ICMPv4TimestampReplyType` class is given in [Table 3-68](#).

Table 3-68. Properties of the `ICMPv4TimestampReplyType` class

Name	Type	Multiplicity	Description
Timestamp_Reply	<code>basicDataTypes:Boolean</code>	0..1	The <code>Timestamp_Reply</code> property specifies whether the subtype of the message is a timestamp reply message (code=14).
Originate_Timestamp	<code>cyboxCommon:UnsignedIntegerObjectPropertyType</code>	0..1	The <code>Originate_Timestamp</code> property specifies the timestamp of the time that the sender last touched the message before sending it. If the time is not available in milliseconds or cannot be provided with respect to midnight UT, then any time can be inserted in a timestamp provided the high order bit of the timestamp is also set to indicate this non-standard value.
Receive_Timestamp	<code>cyboxCommon:UnsignedIntegerObjectPropertyType</code>	0..1	The <code>Receive_Timestamp</code> property specifies the timestamp of the time the echoer first touched the message on receipt. If the time is not available in milliseconds or cannot be provided with respect to midnight UT, then any time can be inserted in a timestamp provided the high order bit of the timestamp is also set to indicate this non-standard value.

Transmit_Timestamp	cyboxCommon: UnsignedIntegerObjectPropertyType	0..1	The <code>Transmit_Timestamp</code> property specifies the timestamp of the time the echoer last touched the message on sending it. If the time is not available in milliseconds or cannot be provided with respect to midnight UT, then any time can be inserted in a timestamp provided the high order bit of the timestamp is also set to indicate this non-standard value.
---------------------------	---	------	--

3.3.4.4.6 ICMPv4AddressMaskRequestType Class

The `ICMPv4AddressMaskRequestType` class specifies the address mask request informational message; ICMP class=17.

The property table of the `ICMPv4AddressMaskRequestType` class is given in [Table 3-69](#).

Table 3-69. Properties of the `ICMPv4AddressMaskRequestType` class

Name	Type	Multiplicity	Description
Address_Mask_Request	<code>basicDataTypes:Boolean</code>	0..1	The <code>Address_Mask_Request</code> property specifies whether the subtype of the message is an address mask request message (code=17).
Address_Mask	<code>AddressObj:</code> <code>AddressObjectType</code>	0..1	The <code>Address_Mask</code> property specifies the address mask. It can be set to zero (as opposed to an address mask reply message, in which case it should be set to the subnet mask).

3.3.4.4.7 ICMPv4AddressMaskReplyType Class

The `ICMPv4AddressMaskReplyType` class specifies the address mask informational message; ICMP class=18.

The property table of the `ICMPv4AddressMaskReplyType` class is given in [Table 3-70](#).

Table 3-70. Properties of the `ICMPv4AddressMaskReplyType` class

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

Address_Mask_Reply	basicDataTypes:Boolean	0..1	The Address_Mask_Reply property specifies whether the subtype of the message is an address mask reply message (code=18).
Address_Mask	AddressObj: AddressObjectType	0..1	This Address_Mask property should be set to the subnet mask.

3.3.4.5 ICMPv4TracerouteType Class

The ICMPv4TracerouteType class specifies properties associated with ICMPv4 traceroute message (class =30). See <http://www.networksorcery.com/enp/protocol/icmp/msg30.htm> for more information.

The property table of the ICMPv4TracerouteType class is given in [Table 3-71](#).

Table 3-71. Properties of the ICMPv4TracerouteType class

Name	Type	Multiplicity	Description
Has_Choice	ICMPv4TracerouteChoiceType	0..1	The Has_Choice property is associated with the class ICMPv4TracerouteChoiceType. It indicates that there is a choice between the Outbound_Packet_Forward_Success property or the Outbound_Packet_no_Route property. Only one of the properties of ICMPv4TracerouteChoiceType class can be populated at any time. See Section 1.2.3 for more detail.
Identifier	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Identifier property specifies the ID number as copied from the ICMP traceroute option of the packet which caused this traceroute message to be sent (not related to the ID number in the IP header).
Outbound_Hop_Count	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Outbound_Hop_Count property specifies the outbound hop count as copied from the IP traceroute option of the

			packet which caused this traceroute message to be sent.
Return_Hop_Count	cyboxCommon: HexBinaryObjectType	0..1	The <code>Return_Hop_Count</code> property specifies the return hop count as copied from the IP traceroute options of the packet which caused this traceroute message to be sent.
Output_Link_Speed	cyboxCommon: HexBinaryObjectType	0..1	The <code>Output_Link_Speed</code> property specifies the speed in bytes per second of the link over which the Outbound/Return Packet will be sent. If this value cannot be determined, the property should be set to zero.
Output_Link_MTU	cyboxCommon: HexBinaryObjectType	0..1	The <code>Output_Link_MTU</code> property specifies the MTU in bytes of the link over which the Outbound/Return Packet will be sent. MTU refers to the data portion (includes IP header; excludes datalink header/trailer) of the packet. If this value cannot be determined, this property should be set to zero.

The `ICMPv4TracerouteChoiceType` class is the type of the `Has_Choice` property. In the UML model, this class is associated with the `<<choice>>` UML stereotype, which specifies that only one of the available properties of the `ICMPv4TracerouteChoiceType` class can be populated at any time. The property table of the `ICMPv4TracerouteChoiceType` class is given in [Table 3-72](#).

Table 3-72. Properties of the `ICMPv4TracerouteChoiceType` class

Name	Type	Multiplicity	Description
Outbound_Packet_Forward_Success	<code>basicDataTypes:Boolean</code>	0..1	The <code>Outbound_Packet_Forward_Success</code> property specifies whether the subtype of the message indicates that the outbound packet was successfully forwarded (<code>code=0</code>). Only one of the <code>Outbound_Packet_Forward_Success</code> and <code>Outbound_Packet_no_Route</code> properties can be populated.

Outbound_Packet_no_Route	basicDataTypes: Boolean	0..1	<p>The Outbound_Packet_no_Route property specifies whether the subtype of the message indicates that there was no route for the outbound packet and the packet was discarded (code=1).</p> <p>Only one of the Outbound_Packet_Forward_Success and Outbound_Packet_no_Route properties can be populated.</p>
---------------------------------	-------------------------	------	---

3.3.5 ICMPv6PacketType Class

The `ICMPv6PacketType` class characterizes the ICMP (v4), which is used send error messages (e.g., a datagram cannot reach its destination), informational messages (e.g., ping). Only the message class defined in <http://tools.ietf.org/html/rfc4443> (ICMP v6) are included; additional message types will be defined as needed. See <http://www.networksorcery.com/enp/protocol/icmpv6.htm> and <http://en.wikipedia.org/wiki/ICMPv6> for more information.

The UML diagram corresponding to the `ICMPv6PacketType` class is shown in [Figure 3-11](#).

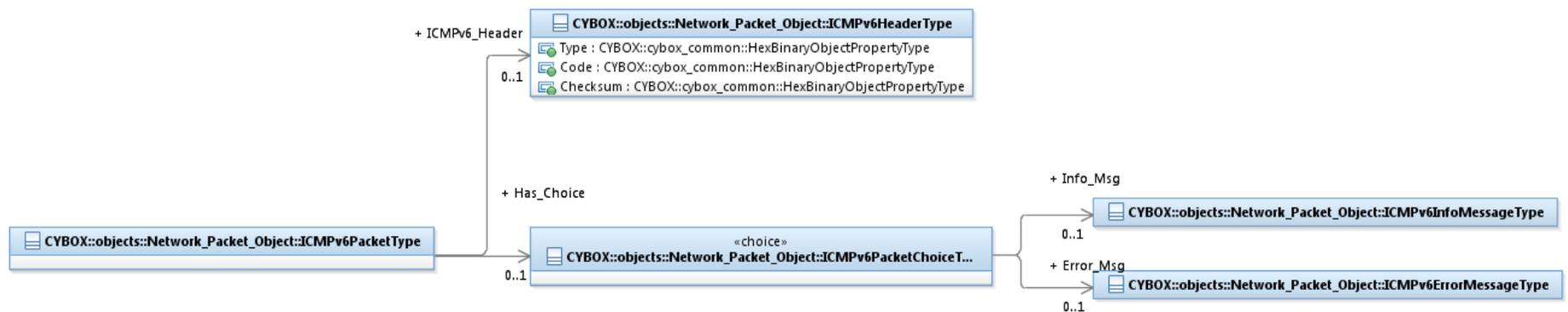


Figure 3-11. UML diagram for the `ICMPv6PacketType` class

The property table of the `ICMPv6PacketType` class is given in [Table 3-73](#).

Table 3-73. Properties of the `ICMPv6PacketType` class

Name	Type	Multiplicity	Description
ICMPv6_Header	ICMPv6HeaderType	0..1	The <code>ICMPv6_Header</code> property specifies the ICMP type, ICMP code, and the checksum.
Has_Choice	ICMPv6PacketChoiceType	0..1	<p>The <code>Has_Choice</code> property is associated with the class <code>ICMPv6PacketChoiceType</code>. It indicates that there is a choice between the <code>Error_Msg</code> property or the <code>Info_Msg</code> property.</p> <p>Only one of the properties of <code>ICMPv6PacketChoiceType</code> class can be populated at any time. See Section 1.2.3 for more detail.</p>

The `ICMPv6PacketChoiceType` class is the type of the `Has_Choice` property. In the UML model, this class is associated with the <<choice>> UML stereotype, which specifies that only one of the available properties of the `ICMPv6PacketChoiceType` class can be populated at any time. The property table of the `ICMPv6PacketChoiceType` class is given in [Table 3-74](#).

Table 3-74. Properties of the `ICMPv6PacketChoiceType` class

Name	Type	Multiplicity	Description
Error_Msg	ICMPv6ErrorMessageType	0..1	<p>The <code>Error_Msg</code> property is used to specify an <code>ICMPv6ErrorMessageType</code> form of the <code>ICMPv6PacketType</code> class.</p> <p>Only one of the <code>Error_Msg</code> and <code>Info_Msg</code> properties can be populated.</p>
Info_Msg	ICMPv6InfoMessageType	0..1	<p>The <code>Info_Msg</code> property is used to specify an <code>ICMPv6InfoMessageType</code> form of the <code>ICMPv6PacketType</code> class.</p> <p>Only one of the <code>Error_Msg</code> and <code>Info_Msg</code> properties can be populated.</p>

3.3.5.1 ICMPv6HeaderType Class

The `ICMPv6HeaderType` class specifies the header properties: the ICMP class, ICMP code, and the checksum.

The property table of the `ICMPv6HeaderType` class is given in [Table 3-75](#).

Table 3-75. Properties of the `ICMPv6HeaderType` class

Name	Type	Multiplicity	Description
Type	<code>cyboxCommon:</code> <code>HexBinaryObjectPropertyType</code>	0..1	The <code>Type</code> property specifies the type of the message. Values range from 0 to 127 (high order bit is 0) indicate an error messages; values from 128 to 255 (high order bit is 1) indicate an informational message.
Code	<code>cyboxCommon:</code> <code>HexBinaryObjectPropertyType</code>	0..1	The <code>Code</code> property depends on the message type and provides an additional level of message granularity.
Checksum	<code>cyboxCommon:</code> <code>HexBinaryObjectPropertyType</code>	0..1	The <code>Checksum</code> property specifies the checksum information of an ICMPv6 header.

3.3.5.2 ICMPv6ErrorMessageType Class

The `ICMPv6ErrorMessageType` class specifies ICMP v6 error messages, including destination unreachable messages, packet too big messages, time exceeded messages, and parameter problem messages. Type values in the header of ICMP v6 error messages range from 1 to 127.

See <http://tools.ietf.org/html/rfc4443> and <http://tools.ietf.org/html/rfc2463> for more information.

The UML diagram corresponding to the `ICMPv6ErrorMessageType` class is shown in [Figure 3-12](#).

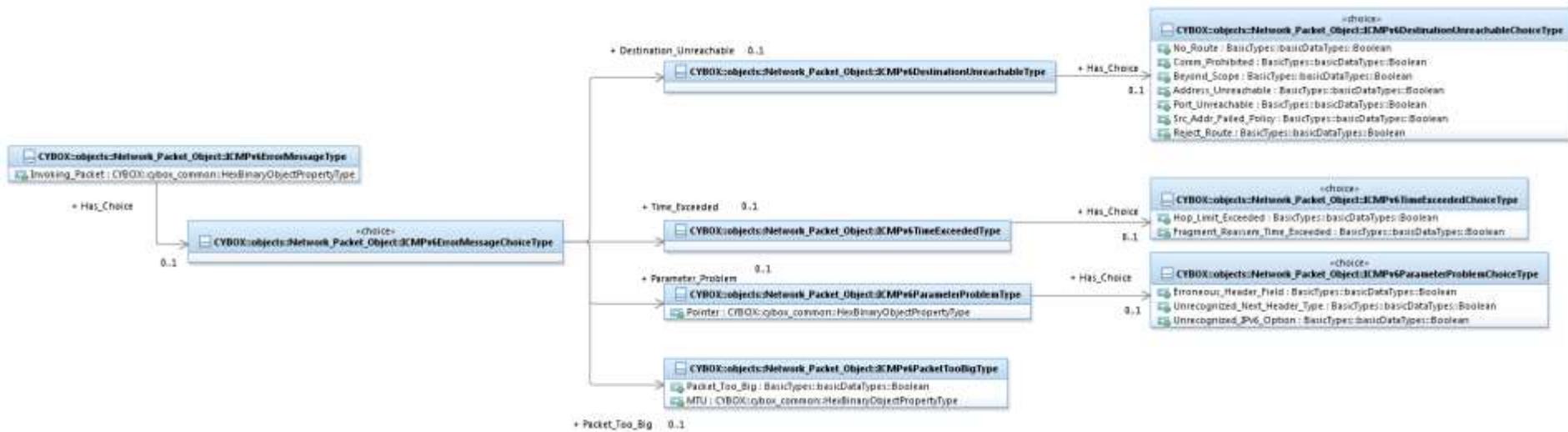


Figure 3-12. UML diagram for the *ICMPv6ErrorMessageType* class

The property table of the *ICMPv6ErrorMessageType* class is given in [Table 3-76](#).

Table 3-76. Properties of the *ICMPv6ErrorMessageType* class

Name	Type	Multiplicity	Description
Invoking_Packet	cyboxCommon: HexBinaryObjectType	0..1	The <i>Invoking_Packet</i> property specifies as much of invoking packet as possible without the <i>ICMPv6</i> packet exceeding the minimum <i>IPv6</i> MTU.
Has_Choice	<i>ICMPv6ErrorMessageChoiceType</i>	0..1	The <i>Has_Choice</i> property is associated with the class <i>ICMPv6ErrorMessageChoiceType</i> . It indicates that there is a choice among <i>Destination_Unreachable</i> , <i>Packet_Too_Big</i> , <i>Time_Exceeded</i> and <i>Parameter_Problem</i> properties.

			Only one of the properties of <code>ICMPv6ErrorMessageChoiceType</code> class can be populated at any time. See Section 1.2.3 for more detail.
--	--	--	--

The `ICMPv6ErrorMessageChoiceType` class is the type of the `Has_Choice` property. In the UML model, this class is associated with the `<<choice>>` UML stereotype, which specifies that only one of the available properties of the `ICMPv6ErrorMessageChoiceType` class can be populated at any time. The property table of the `ICMPv6ErrorMessageChoiceType` class is given in Table 3-77.

Table 3-77. Properties of the `ICMPv6ErrorMessageChoiceType` class

Name	Type	Multiplicity	Description
Destination_Unreachable	<code>ICMPv6DestinationUnreachableType</code>	0..1	<p>The <code>Destination_Unreachable</code> property specifies a destination unreachable message. It should be generated by a router, or by the IPv6 later in the originating node, in response to a packet that cannot be delivered to its destination address for reasons other than congestion.</p> <p>Only one of the <code>Destination_Unreachable</code>, <code>Packet_Too_Big</code>, <code>Time_Exceeded</code> and <code>Parameter_Problem</code> properties can be populated.</p>
Packet_Too_Big	<code>ICMPv6PacketTooBigType</code>	0..1	<p>The <code>Packet_Too_Big</code> property specifies a packet too big message. It must be sent by a router in response to a packet that it cannot forward because the packet is larger than the MTU of the outgoing link.</p> <p>Only one of the <code>Destination_Unreachable</code>, <code>Packet_Too_Big</code>, <code>Time_Exceeded</code> and <code>Parameter_Problem</code> properties can be populated.</p>
Time_Exceeded	<code>ICMPv6TimeExceededType</code>	0..1	The <code>Time_Exceeded</code> property specifies a time

			<p>exceeded message. It is sent if either the hop limit is exceeded (hop limit = 0) or if fragment reassembly has timed out.</p> <p>Only one of the <code>Destination_Unreachable</code>, <code>Packet_Too_Big</code>, <code>Time_Exceeded</code> and <code>Parameter_Problem</code> properties can be populated.</p>
Parameter_Problem	<code>ICMPv6ParameterProblemType</code>	0..1	<p>The <code>Parameter_Problem</code> property specifies a parameter problem message. If an IPv6 node processing a packet finds a problem with a property in the IPv6 header or extension headers and it cannot complete processing the packet, it should send an ICMPv6 Parameter Problem message to the packet's source.</p> <p>Only one of the <code>Destination_Unreachable</code>, <code>Packet_Too_Big</code>, <code>Time_Exceeded</code> and <code>Parameter_Problem</code> properties can be populated.</p>

3.3.5.2.1 ICMPv6DestinationUnreachableType Class

The `ICMPv6DestinationUnreachableType` class specifies a destination unreachable error message; ICMP v6 class=1.

In CybOX 2.1.1, all of the properties of the `ICMPv6DestinationUnreachableType` class are mutually exclusive, i.e., only one property can be populated. This restriction is based on the fact each property is a Boolean value to indicate if the message is the one of seven types of Destination Unreachable messages. In future releases, this could be modelled as using one property for the subtype (code) and an enumeration.

The property table of the `ICMPv6DestinationUnreachableType` class is given in [Table 3-78](#).

Table 3-78. Properties of the `ICMPv6DestinationUnreachableType` class

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

No_Route	basicDataTypes:Boolean	0..1	The No_Route property indicates whether no route to destination exists (ICMP v6 code=0).
Comm_Prohibited	basicDataTypes:Boolean	0..1	The Comm_Prohibited property indicates whether communication with destination is administratively prohibited (ICMP v6 code=1).
Beyond_Scope	basicDataTypes:Boolean	0..1	The Beyond_Scope property indicates whether destination is beyond the scope of source address (ICMP v6 code =2).
Address_Unreachable	basicDataTypes:Boolean	0..1	The Address_Unreachable property indicates whether an address is unreachable (ICMP v6 code=3).
Port_Unreachable	basicDataTypes:Boolean	0..1	The Port_Unreachable property indicates whether a port is unreachable (ICMP v6 code=4).
Src_Addr_Failed_Policy	basicDataTypes:Boolean	0..1	The Src_Addr_Failed_Policy property indicates whether a source address failed the ingress/egress policy (ICMP v6 code=5).
Reject_Route	basicDataTypes:Boolean	0..1	The Reject_Route property indicates whether the route to destination should be rejected (ICMP v6 code=6).

3.3.5.2.2 ICMPv6PacketTooBigType Class

The `ICMPv6PacketTooBigType` class specifies the packet too big error message; ICMP v6 class=2.

The property table of the `ICMPv6PacketTooBigType` class is given in [Table 3-79](#).

Table 3-79. Properties of the `ICMPv6PacketTooBigType` class

Name	Type	Multiplicity	Description
Packet_Too_Big	basicDataTypes:Boolean	0..1	The Packet_Too_Big property is set to 0 (zero) by the originator

			and ignored by the receiver.
MTU	cyboxCommon: HexBinaryObjectType	0..1	The MTU property specifies the Maximum Transmission Unit (MTU) size limit.

3.3.5.2.3 ICMPv6TimeExceededType Class

The `ICMPv6TimeExceededType` class specifies the time exceeded error message; ICMP v6 class=3. The UML diagram corresponding to the `ICMPv6TimeExceededType` class is shown in [Figure 3-13](#).

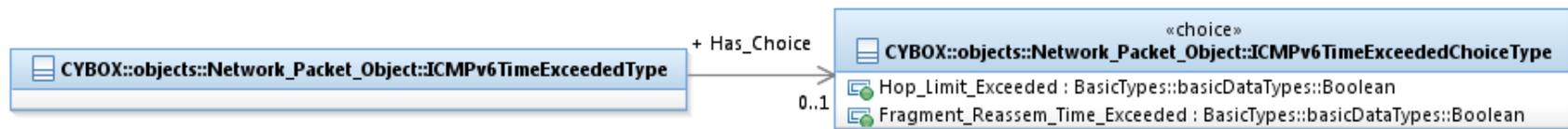


Figure 3-13. UML diagram for the `ICMPv6TimeExceededType` class

The property table of the `ICMPv6TimeExceededType` class is given in [Table 3-80](#).

Table 3-80. Properties of the `ICMPv6TimeExceededType` class

Name	Type	Multiplicity	Description
Has_Choice	<code>ICMPv6TimeExceededChoiceType</code>	0..1	<p>The <code>Has_Choice</code> property is associated with the class <code>ICMPv6TimeExceededChoiceType</code>. It indicates that there is a choice between the <code>Hop_Limit_Exceeded</code> property or the <code>Fragment_Reassem_Time_Exceeded</code> property.</p> <p>Only one of the properties of <code>ICMPv6TimeExceededChoiceType</code> class can be populated at any time. See Section 1.2.3 for more detail.</p>

The `ICMPv6TimeExceededChoiceType` class is the type of the `Has_Choice` property. In the UML model, this class is associated with the <<choice>> UML stereotype, which specifies that only one of the available properties of the `ICMPv6TimeExceededChoiceType` class can be populated at any time. The property table of the `ICMPv6TimeExceededChoiceType` class is given in [Table 3-81](#).

Table 3-81. Properties of the `ICMPv6TimeExceededChoiceType` class

Name	Type	Multiplicity	Description
Hop_Limit_Exceeded	basicDataTypes: Boolean	0..1	The <code>Hop_Limit_Exceeded</code> property specifies whether the hop limit was exceeded in transit (ICMP v6 code=0). Only one of the <code>Hop_Limit_Exceeded</code> and <code>Fragment_Reassem_Time_Exceeded</code> properties can be populated.
Fragment_Reassem_Time_Exceeded	basicDataTypes :Boolean	0..1	The <code>Fragment_Reassem_Time_Exceeded</code> property specifies whether the fragment reassembly time was exceeded (ICMP v6 code=1). Only one of the <code>Hop_Limit_Exceeded</code> and <code>Fragment_Reassem_Time_Exceeded</code> properties can be populated.

3.3.5.2.4 ICMPv6ParameterProblemType Class

The `ICMPv6ParameterProblemType` class specifies a parameter problem error message; ICMP v6 class=4. The UML diagram corresponding to the `ICMPv6ParameterProblemType` class is shown in [Figure 3-14](#).

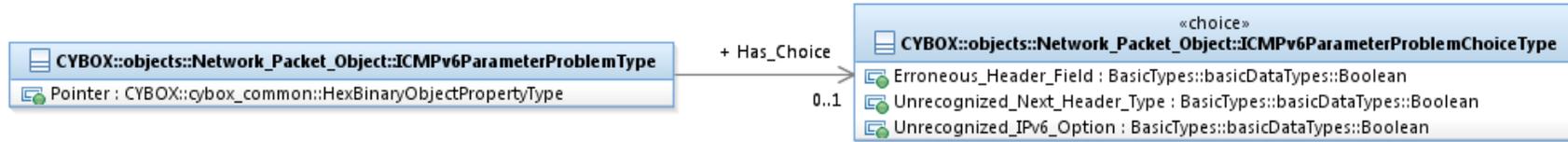


Figure 3-14. UML diagram of the *ICMPv6ParameterProblemType* class

The property table of the *ICMPv6ParameterProblemType* class is given in [Table 3-82](#).

Table 3-82. Properties of the *ICMPv6ParameterProblemType* class

Name	Type	Multiplicity	Description
Pointer	cyboxCommon: HexBinaryObjectType	0..1	The <code>Pointer</code> property specifies the octet offset within invoking packet where error was detected.
Erroneous_Header_Field	basicDataTypes:Boolean	0..1	The <code>Erroneous_Header_Field</code> property indicates whether an erroneous header field was encountered (ICMP v6 code=0). Only one of the <code>Erroneous_Header_Field</code> , <code>Unrecognized_Next_Header_Type</code> and <code>Unrecognized_IPv6_Option</code> properties can be populated.
Unrecognized_Next_Header_Type	basicDataTypes:Boolean	0..1	The <code>Unrecognized_Next_Header_Type</code> property indicates whether an unrecognized next header type was encountered (ICMP v6 code=1). Only one of the <code>Erroneous_Header_Field</code> , <code>Unrecognized_Next_Header_Type</code> and <code>Unrecognized_IPv6_Option</code> properties can be populated.

Unrecognized_IPv6_Option	basicDataTypes:Boolean	0..1	<p>The Unrecognized_IPv6_Option property indicates whether an unrecognized IP v6 option was encountered (ICMP v6 code=2).</p> <p>Only one of the Erroneous_Header_Field, Unrecognized_Next_Header_Type and Unrecognized_IPv6_Option properties can be populated.</p>
---------------------------------	------------------------	------	--

3.3.5.3 ICMPv6InfoMessageType Class

The `ICMPv6InfoMessageType` class specifies ICMP v6 informational messages include echo request/reply; other informational message class will be added in the future as they are more commonly used (only echo request/reply are defined in <http://tools.ietf.org/html/rfc4443>). The UML diagram corresponding to the `ICMPv6InfoMessageType` class is shown in **Figure 3-15**.



Figure 3-15. UML diagram for `ICMPv6InfoMessageType` class

The property table of the `ICMPv6InfoMessageType` class is given in **Table 3-83**.

Table 3-83. Properties of the `ICMPv6InfoMessageType` class

Name	Type	Multiplicity	Description
Info_Msg_Content	<code>ICMPv6InfoMessageContentType</code>	0..1	The <code>Info_Msg_Content</code> property specifies properties that are common to all ICMP v6 informational messages. Properties that

			are specific to individual messages are defined separately under each message type.
Has_Choice	ICMPv6InfoMessageChoiceType	0..1	<p>The <code>Has_Choice</code> property is associated with the class <code>ICMPv6InfoMessageChoiceType</code>. It indicates that there is a choice between the <code>Echo_Reply</code> and <code>Echo_Request</code> properties .</p> <p>Only one of the properties of <code>ICMPv6InfoMessageChoiceType</code> class can be populated at any time. See Section 1.2.3 for more detail.</p>

The `ICMPv6InfoMessageChoiceType` class is the type of the `Has_Choice` property. In the UML model, this class is associated with the `<<choice>>` UML stereotype, which specifies that only one of the available properties of the `ICMPv6InfoMessageChoiceType` class can be populated at any time. The property table of the `ICMPv6InfoMessageChoiceType` class is given in [Table 3-84](#).

Table 3-84. Properties of the `ICMPv6InfoMessageChoiceType` class

Name	Type	Multiplicity	Description
Echo_Request	ICMPv6EchoRequestType	0..1	<p>The <code>Echo_Request</code> property specifies an echo request message (type=128). These messages are also known as "ping".</p> <p>Only one of the <code>Echo_Reply</code> and <code>Echo_Request</code> properties can be populated.</p>
Echo_Reply	ICMPv6EchoReplyType	0..1	<p>The <code>Echo_Reply</code> property specifies an echo reply message (type=129). These messages are also known as "ping".</p> <p>Only one of the <code>Echo_Reply</code> and <code>Echo_Request</code> properties can be populated.</p>

3.3.5.3.1 ICMPv6InfoMessageContentType Class

The `ICMPv6InfoMessageContentType` class specifies properties common to the ICMPv6 informational messages.

The property table of the `ICMPv6InfoMessageContentType` class is given in [Table 3-85](#).

Table 3-85. Properties of the `ICMPv6InfoMessageContentType` class

Name	Type	Multiplicity	Description
Identifier	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>Identifier</code> property specifies a 16-bit identifier, which is combined with the sequence number, and is called the "quench" for echo reply and echo request.
Sequence_Number	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>Sequence_Number</code> property specifies a 16-bit sequence number. The identifier and sequence number can be used by the client to match the reply with the request that caused the reply.

3.3.5.3.2 ICMPv6EchoRequestType Class

The `ICMPv6EchoRequestType` class specifies an echo request informational ICMP v6 message; class=128.

The property table of the `ICMPv6EchoRequestType` class is given in [Table 3-86](#).

Table 3-86. Properties of the `ICMPv6EchoRequestType` class

Name	Type	Multiplicity	Description
Echo_Request	basicDataTypes:Boolean	0..1	The <code>Echo_Request</code> property specifies whether the subtype of the message is an echo request message (code=128).
Data	cyboxCommon: HexBinaryObjectPropertyType	0..1	The <code>Data</code> property specifies zero or more octets of arbitrary data.

3.3.5.3.3 ICMPv6EchoReplyType Class

Echo reply informational ICMP v6 message; class=129.

The property table of the `ICMPv6EchoReplyType` class is given in [Table 3-87](#).

Table 3-87. Properties of the `ICMPv6EchoReplyType` class

Name	Type	Multiplicity	Description
Echo_Reply	<code>basicDataTypes:Boolean</code>	0..1	The <code>Echo_Reply</code> property specifies whether the subtype of the message is an echo reply message (code=129).
Data	<code>cyboxCommon:HexBinaryObjectType</code>	0..1	The <code>Data</code> property specifies the data resulting from the invoking echo request message.

3.4 TransportLayerType Class

The `TransportLayerType` class specifies the properties of the UDP and TCP protocol. Other protocols will be defined as necessary. The UML diagram corresponding to the `TransportLayerType` class is shown in [Figure 3-1](#).

The property table of the `TransportLayerType` class is given in [Table 3-88](#).

Table 3-88. Properties of the `TransportLayerType` class

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

The `TransportLayerChoiceType` class is the type of the `Has_Choice` property. In the UML model, this class is associated with the `<<choice>>` UML stereotype, which specifies that only one of the available properties of the `TransportLayerChoiceType` class can be populated at any time. The property table of the `TransportLayerChoiceType` class is given in [Table 3-89](#).

Table 3-89. Properties of the `TransportLayerChoiceType` class

Name	Type	Multiplicity	Description
TCP	<code>TCPType</code>	0..1	<p>The <code>TCP</code> property specifies a TCP packet. TCP provides reliable, ordered delivery of a stream of bytes from a program on one computer to another program on another computer.</p> <p>Only one of the <code>TCP</code> and <code>UDP</code> properties can be populated.</p>
UDP	<code>UDPType</code>	0..1	<p>The <code>UDP</code> property specifies a UDP packet. UDP uses a simple transmission model without implicit handshaking dialogues for providing reliability, ordering, or data integrity. Thus, UDP provides an unreliable service and datagrams may arrive out of order, appear duplicated, or go missing without notice.</p> <p>Only one of the <code>TCP</code> and <code>UDP</code> properties can be populated.</p>

3.4.1 TCPType Class

The `TCPType` class specifies a TCP packet. The TCP protocol provides reliable, ordered delivery of a stream of bytes from a program on one computer to another program on another computer. See http://en.wikipedia.org/wiki/Transmission_Control_Protocol for more information.

The UML diagram corresponding to the `TCPType` class is shown in [Figure 3-16](#).



Figure 3-16. UML diagram of the *TCPTType* class

The property table of the *TCPTType* class is given in [Table 3-90](#).

Table 3-90. Properties of the *TCPTType* class

Name	Type	Multiplicity	Description
TCP_Header	TCPHeaderType	0..1	The <i>TCP_Header</i> property specifies the 10 mandatory properties and an optional extension property.
Options	cyboxCommon: HexBinaryObjectType	0..1	The <i>Option</i> property contains the TCP Options. There are up to three properties: Option-Kind (1 byte), Option-Length (1 byte), Option-Data (variable). This property will be further defined when required.
Data	cyboxCommon:DataSegmentType	0..1	The <i>Data</i> property specifies the data payload of the TCP packet.

3.4.1.1 TCPHeaderType Class

The *TCPHeaderType* class contains 10 mandatory fields and an optional extension field.

The property table of the *TCPHeaderType* class is given in [Table 3-91](#).

Table 3-91. Properties of the *TCPHeaderType* class

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

Src_Port	PortObj:PortObjectType	0..1	The Src_Port property specifies the sending port.
Dest_Port	PortObj:PortObjectType	0..1	The Dest_Port property specifies the receiving port.
Seq_Num	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Seq_Num property specifies the Sequence number (32-bits). The Sequence number has a dual role: If the <code>syn</code> flag is set, then this is the initial sequence numbers. If the <code>syn</code> flag is clear (see <code>TCP_Flags</code> property), then this is the accumulated sequence number of the first data byte of this packet for the current session.
ACK_Num	cyboxCommon: HexBinaryObjectPropertyType	0..1	The ACK_Num property specifies whether the <code>ack</code> flag (see <code>TCP_Flags</code> property) is set then the value of this property is the next sequence number that the receiver is expecting.
Data_Offset	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Data_Offset property specifies the size of the TCP header in 32-bit words.
Reserved	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Reserved property specifies that these 3 bits are reserved for future use and should be set to zero.
TCP_Flags	TCPFlagsType	0..1	The TCP_Flags property specifies the 9 flags (aka Control Bits).
Window	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Window property specifies the size of the receive window, which specifies the number of bytes (beyond the sequence number in the acknowledgment property) that the sender of this segment is currently willing to receive.
Checksum	cyboxCommon: HexBinaryObjectPropertyType	0..1	The Checksum property specifies the 16-bit checksum which is used for error-checking of the header and data.
Urg_Ptr	cyboxCommon:	0..1	The Urg_Ptr property specifies a 16-bit property that is an offset from the sequence number indicating the last urgent data byte when the <code>urg</code>

	HexBinaryObjectType		flag is set
--	---------------------	--	-------------

3.4.1.2 TCPFlagsType Class

The `TCPFlagsType` class specifies the nine different flags in the TCP header.

The property table of the `TCPFlagsType` class is given in [Table 3-92](#).

Table 3-92. Properties of the `TCPFlagsType` class

Name	Type	Multiplicity	Description
ns	<code>basicDataTypes:Boolean</code>	0..1	The <code>ns</code> property specifies the ECN-nonce concealment protection setting.
cwr	<code>basicDataTypes:Boolean</code>	0..1	The <code>cwr</code> property specifies whether the Congestion Window Reduced (CWR) flag is set by the sending host to indicate that it received a TCP segment with the ECE flag set and had responded in congestion control mechanism.
ece	<code>basicDataTypes:Boolean</code>	0..1	The <code>ece</code> property specifies the ECN-Echo flag has a dual role: if the <code>syn</code> flag is set, the TCP peer is ECN capable; if the <code>syn</code> flag is clear, a packet with Congestion Experienced flag set (ECN=11) in IP header is received during normal transmission (see http://tools.ietf.org/html/rfc3168).
urg	<code>basicDataTypes:Boolean</code>	0..1	The <code>urg</code> property specifies whether the <code>Urg_Ptr</code> property is significant.
ack	<code>basicDataTypes:Boolean</code>	0..1	The <code>ack</code> property specifies whether the <code>Acknowledgment</code> property is significant. All packets after the initial <code>syn</code> packet sent by the client should have this flag set.
psh	<code>basicDataTypes:Boolean</code>	0..1	The <code>psh</code> property specifies whether to push the buffered data to the receiving application.
rst	<code>basicDataTypes:Boolean</code>	0..1	The <code>rst</code> property specifies whether to reset the connection.

syn	basicDataTypes:Boolean	0..1	The <code>syn</code> property specifies whether to synchronize sequence numbers. Only the first packet sent from each end should have this flag set.
fin	basicDataTypes:Boolean	0..1	The <code>fin</code> property specifies whether there is no more data from sender.

3.4.2 UDPTYPE Class

The `UDPTYPE` class specifies a UDP packet. The UDP protocol uses a simple transmission model without implicit handshaking dialogues for providing reliability, ordering, or data integrity. Thus, UDP provides an unreliable service and datagrams may arrive out of order, appear duplicated, or go missing without notice. See http://en.wikipedia.org/wiki/User_Datagram_Protocol for more information.

The UML diagram corresponding to the `UDPTYPE` class is shown in **Figure 3-17**.

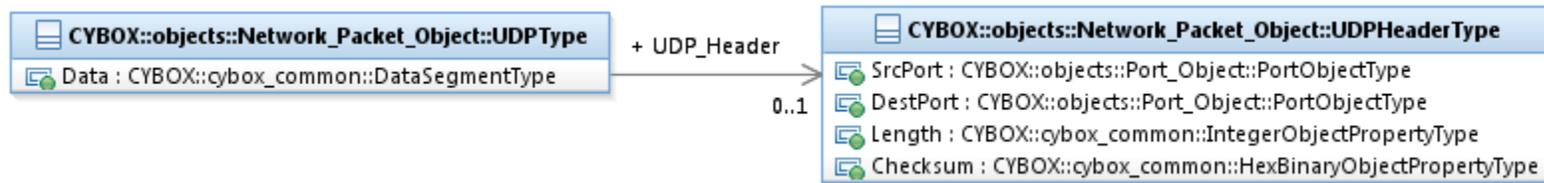


Figure 3-17. UML diagram for `UDPTYPE` class

The property table of the `UDPTYPE` class is given in **Table 3-93**.

Table 3-93. Properties of the `UDPTYPE` class

Name	Type	Multiplicity	Description
UDP_Header	UDPHeaderType	0..1	The <code>UDP_Header</code> properties consists of four properties.
Data	cyboxCommon:DataSegmentType	0..1	The <code>Data</code> property specifies the data payload of the UDP packet.

3.4.2.1 UDPHeaderType Class

The UDP header class defines the four fields in the UDP header.

The property table of the `UDPHeaderType` class is given in [Table 3-94](#).

Table 3-94. Properties of the `UDPHeaderType` class

Name	Type	Multiplicity	Description
SrcPort	<code>PortObj:PortObjectType</code>	0..1	The <code>SrcPort</code> property specifies the sender's port.
DestPort	<code>PortObj:PortObjectType</code>	0..1	The <code>DestPort</code> property specifies the receiver's port.
Length	<code>cyboxCommon:IntegerObjectPropertyType</code>	0..1	The <code>Length</code> property specifies the length in bytes of the entire datagram (header and data).
Checksum	<code>cyboxCommon:HexBinaryObjectPropertyType</code>	0..1	The <code>Checksum</code> property specifies the checksum is used for error-checking of the header and data.

3.5 IANAPortNumberRegistryType Data Type²

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

3.6 Enumerations

3.6.1 ARPOpTypeEnum Enumeration

The literals of the `ARPOpTypeEnum` enumeration are given in [Table 3-95](#).

Table 3-95. Literals of the `ARPOpTypeEnum` enumeration

Enumeration Literal	Description
ARP request(1)	Indicates the ARP request operation, or value 1 in the OPER field of an ARP packet.
ARP reply(2)	Indicates the ARP reply operation, or value 2 in the OPER field of an ARP packet.
RARP request(3)	Indicates the RARP request operation, or value 3 in the OPER field of an ARP packet.
RARP reply(4)	Indicates the RARP reply operation, or value 4 in the OPER field of an ARP packet.

3.6.2 DoNotFragmentTypeEnum Enumeration

The literals of the `DoNotFragmentTypeEnum` enumeration are given in [Table 3-96](#).

Table 3-96. Literals of the `DoNotFragmentTypeEnum` enumeration

Enumeration Literal	Description
fragmentifnecessary(0)	Indicates that the router or other device should fragment the packet if necessary, especially if the packet size is bigger than the MTU of an outgoing interface.
donotfragment(1)	Indicates that the router or other device should NOT fragment the packet in any circumstance.

3.6.3 MoreFragmentsTypeEnum Enumeration

The literals of the `MoreFragmentsTypeEnum` enumeration are given in [Table 3-97](#).

Table 3-97. Literals of the `MoreFragmentsTypeEnum` enumeration

Enumeration Literal	Description
<code>lastfragment(0)</code>	Indicates that the last fragment has been received. In other words, the "more fragments" flag is set to 0.
<code>morefragmentstofollow(1)</code>	Indicates that more fragments need to be received. In other words, the "more fragments" flag is set.

3.6.4 IPv4CopyFlagTypeEnum Enumeration

The literals of the `IPv4CopyFlagTypeEnum` enumeration are given in [Table 3-98](#).

Table 3-98. Literals of the `IPv4CopyFlagTypeEnum` enumeration

Enumeration Literal	Description
<code>donotcopy(0)</code>	Indicates that the options need NOT be copied into all fragments of a fragmented packet.
<code>copy(1)</code>	Indicates that the options need to be copied into all fragments of a fragmented packet.

3.6.5 IPv4ClassTypeEnum Enumeration

The literals of the `IPv4ClassTypeEnum` enumeration are given in [Table 3-99](#).

Table 3-99. Literals of the `IPv4ClassTypeEnum` enumeration

Enumeration Literal	Description
control(0)	Indicates the "control" options.
reserved(1)	Indicates a reserved value.
debuggingandmeasurement(2)	Indicates the debugging and measurement options.
reserved(3)	Indicates a reserved value.

3.6.6 IPv4OptionsTypeEnum Enumeration

The literals of the `IPv4OptionsTypeEnum` enumeration are given in [Table 3-100](#).

Table 3-100. Literals of the `IPv4OptionsTypeEnum` enumeration

Enumeration Literal	Description
endofoptionslist(0)	Indicates the End of Options List option, or EOOL.
nop(1)	Indicates the No Operation option, or NOP.
security(2)	Indicates the Security option, or SEC.
loosesourceroute(3)	Indicates the Loose Source Route option, or LSR.
timestamp(4)	Indicates the Time Stamp option, or TS.
extendedsecurity(5)	Indicates the Extended Security option, or E-SEC.

commercialsecurity(6)	Indicates the Commercial Security option, or CIPSO.
recordroute(7)	Indicates the Record Route option, or RR.
streamidentifier(8)	Indicates the Stream ID option, or SID.
strictsourceroute(9)	Indicates the Strict Source Route option, or SSR.
experimentalmeasure(10)	Indicates the Experimental Measurement option, or ZSU.
mtuprobe(11)	Indicates the MTU probe option, or MTUP.
mtureply(12)	Indicates the MTU reply option, or MTUR.
experimentalflowcontrol(13)	Indicates the Experimental Flow Control option, or FINN.
experimentalaccesscontrol(14)	Indicates the Experimental Access Control option, or FINN.
encode(15)	
imitrafficedescriptor(16)	Indicates the IMI Traffic Descriptor option, or IMITD.
extendedip(17)	Indicates the Extended Internet Protocol option, or EIP.
traceroute(18)	Indicates the Trace Route option, or TR.
addressextension(19)	Indicates the Address Extension option, or ADDEXT.
routeralert(20)	Indicates a Router Alert option, or RTRALT.

selectivedirectedbroadcastmode(21)	Indicates a Selective Directed Broadcast option, or SDB.
dynamicpacketstate(23)	Indicates the Dynamic Packet State option, or DPS.
upstreammulticastpacket(24)	Indicates the Upstream Multicast Packet option, or UMP.
quickstart(25)	Indicates the Quick-Start option, or QS.
exp(30)	Indicates the RFC3692-style Experiment option, or EXP.

3.6.7 IPv6DoNotRecogActionTypeEnum Enumeration

The literals of the `IPv6DoNotRecogActionTypeEnum` enumeration are given in [Table 3-101](#).

See <http://tools.ietf.org/html/rfc2460> for more information.

Table 3-101. Literals of the `IPv6DoNotRecogActionTypeEnum` enumeration

Enumeration Literal	Description
skiptoption(00)	Indicates that the option should be skipped and the header should continue to be processed.
discardpacket(01)	Indicates that the packet should be discarded.
discardpacketsendicmpcode2(10)	Indicates that the packet should be discarded and regardless of whether or not the packet's Destination Address was a multicast address, send an ICMP Parameter Problem, Code 2, message to the packet's Source Address, pointing to the unrecognized Option Type.

discardpacketsendicmpcode2nomulti(11)	Indicates that the packet should be discarded and only if the packet's Destination Address was not a multicast address, send an ICMP Parameter Problem, Code 2, message to the packet's Source Address, pointing to the unrecognized Option
--	---

3.6.8 IPv6PacketChangeTypeEnum Enumeration

The literals of the `IPv6PacketChangeTypeEnum` enumeration are given in [Table 3-102](#).

See <http://tools.ietf.org/html/rfc2460> for more information.

Table 3-102. Literals of the `IPv6PacketChangeTypeEnum` enumeration

Enumeration Literal	Description
nochange(0)	Indicates that the packet does not change en-route.
change(1)	Indicates that the packet may change en-route.

3.6.9 IPVersionTypeEnum Enumeration

The literals of the `IPVersionTypeEnum` enumeration are given in [Table 3-103](#).

Table 3-103. Literals of the `IPVersionTypeEnum` enumeration

Enumeration Literal	Description
IPv4(4)	Indicates IP Version 4.

ST(5)	Indicates the IP version designating ST Datagram Mode.
IPv6(6)	Indicates IP Version 6.
TP/IX(7)	Indicates the IP version designating TP/IX: The Next Internet.
PIP(8)	Indicates the IP version designating PIP: The P Internet Protocol.
TUBA(9)	Indicates the IP version designating TUBA (TCP and UDP with Bigger Addresses, i.e. http://tools.ietf.org/html/rfc1347).

3.6.10 IANAHardwareTypeEnum Enumeration

The literals of the `IANAHardwareTypeEnum` enumeration are given in [Table 3-104](#).

Table 3-104. Literals of the `IANAHardwareTypeEnum` enumeration

Enumeration Literal	Description
Ethernet(1)	Indicates Ethernet hardware.
IEEE802(6)	Indicates IEEE 802 compliant hardware for networks carrying variable-size packets.
ARCNET(7)	Indicates the ARCNET LAN protocol.
FrameRelay(15)	Indicates the Frame Relay WAN technology.
ATM(16)	Indicates the ATM (Asynchronous Transfer Mode) networking standard.

HDLC(17)	Indicates the HDLC (High-Level Data Link Control) protocol.
FibreChannel(18)	Indicates the FibreChannel technology.
ATM(19)	Indicates the ATM (Asynchronous Transfer Mode) networking standard.
SerialLine(20)	Indicates the Serial Line protocol, or SLIP.

3.6.11 IANAEtherTypeEnum Enumeration

The literals of the `IANAEtherTypeEnum` enumeration are given in [Table 3-105](#).

Table 3-105. Literals of the `IANAEtherTypeEnum` enumeration

Enumeration Literal	Description
IPv4(0x0800)	Indicates the IPv4 Ethernet type is specified.
ARP(0x0806)	Indicates the ARP Ethernet type is specified.
RARP(0x8035)	Indicates the RARP Ethernet type is specified.
IPX(0x8137)	Indicates the IPX Ethernet type is specified.
SNMP(0x814C)	Indicates the SNMP Ethernet type is specified.
IPv6(0x86DD)	Indicates the IPv6 Ethernet type is specified.

3.6.12 IANAAssignedIPNumbersTypeEnum Enumeration

The literals of the `IANAAssignedIPNumbersTypeEnum` enumeration are given in [Table 3-106](#).

Table 3-106. Literals of the `IANAAssignedIPNumbersTypeEnum` enumeration

Enumeration Literal	Description
IPv6hopbyhop(0)	Indicates the IPv6 Hop-By-Hop option protocol (HOPOPT).
ICMP(1)	Indicates the Internet Control Message protocol (HOPOPT).
IGMP(2)	Indicates the Internet Group Message protocol (HOPOPT).
GGP(3)	Indicates the Gateway-to-Gateway protocol (HOPOPT).
IPv4Encapsulation(4)	Indicates the IPv4 Encapsulation protocol (IPv4).
ST(5)	Indicates the Stream protocol (HOPOPT).
TCP(6)	Indicates the TCP protocol.
EGP(8)	Indicates the EGP (Exterior Gateway) protocol.
IGRP(9)	Indicates the IGP/IGRP (Cisco) protocol.
NVP(11)	Indicates the Network-Voice protocol.
PUP(12)	Indicates the PUP protocol.
ARGUS(13)	Indicates the ARGUS protocol.

EMCON(14)	Indicates the EMCON protocol.
XNET(15)	Indicates the Cross Net Debugger protocol.
UDP(17)	Indicates the UDP protocol.
IPv6Encapsulation(41)	Indicates the IPv6 protocol.
SDRP(42)	Indicates the Source Demand Routing protocol.
IPv6routingheader(43)	Indicates the routing header for IPv6.
IPv6fragmentheader(44)	Indicates the fragment header for IPv6.
RSVP(46)	Indicates the Reservation Protocol.
GRE(47)	Indicates the General Routing Encapsulation protocol number.
encapsultaesecuritypayload_ESP(50)	Indicates the Encapsulated Security Payload protocol number.
authenticationheader_AH(51)	Indicates the Authentication Header protocol number.
ICMPv6(58)	Indicates the ICMP for v6 protocol number.
IPv6nonextheader(59)	Indicates the No Next Header for IPv6 protocol number.
IPv6destinationoptions(60)	Indicates the Destination Options for IPv6 protocol number.

mobilityheader(135)	Indicates the Mobility Header protocol number.
----------------------------	--

3.6.13 IANAPortNumberRegistryTypeEnum Enumeration

The literals of the `IANAPortNumberRegistryTypeEnum` enumeration are given in [Table 3-107](#).

Table 3-107. Literals of the `IANAPortNumberRegistryTypeEnum` enumeration

Enumeration Literal	Description
ftpdata(20)	Indicates the port for ftpdata.
ftp(21)	Indicates the port for ftp.
ssh(22)	Indicates the port for ssh.
telnet(23)	Indicates the port for telnet.
smtp(25)	Indicates the port for smtp.
domain(53)	Indicates the domain port.
tftp(69)	Indicates the port for tftp.
http(80)	Indicates the port for http.
ldap(389)	Indicates the port for ldap.
https(443)	Indicates the port for https.

3.6.14 MFlagTypeEnum Enumeration

The literals of the MFlagTypeEnum enumeration are given in [Table 3-108](#).

Table 3-108. Literals of the MFlagTypeEnum enumeration

Enumeration Literal	Description
lastfragment(0)	Fragment is the last fragment.
morefragments(1)	There are more fragments (current is not the last).

4 Conformance

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

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

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

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

Appendix A. Acknowledgments

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

Aetna

David Crawford

AIT Austrian Institute of Technology

Roman Fiedler

Florian Skopik

Australia and New Zealand Banking Group (ANZ Bank)

Dean Thompson

Blue Coat Systems, Inc.

Owen Johnson

Bret Jordan

Century Link

Cory Kennedy

CIRCL

Alexandre Dulaunoy

Andras Iklody

Raphaël Vinot

Citrix Systems

Joey Peloquin

Dell

Will Urbanski

Jeff Williams

DTCC

Dan Brown

Gordon Hundley

Chris Koutras

EMC

Robert Griffin

Jeff Odom

Ravi Sharda

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

David Eilken

Chris Ricard

Fortinet Inc.

Gavin Chow

Airbus Group SAS

Joerg Eschweiler

Marcos Orallo

Anomali

Ryan Clough

Wei Huang

Hugh Njemanze

Katie Pelusi

Aaron Shelmire

Jason Trost

Bank of America

Alexander Foley

Center for Internet Security (CIS)

Sarah Kelley

Check Point Software Technologies

Ron Davidson

Cisco Systems

Syam Appala

Ted Bedwell

David McGrew

Pavan Reddy

Omar Santos

Jyoti Verma

Cyber Threat Intelligence Network, Inc. (CTIN)

Doug DePeppe

Jane Ginn

Ben Othman

DHS Office of Cybersecurity and Communications (CS&C)

Richard Struse

Marlon Taylor

EclecticIQ

Marko Dragoljevic

Joep Gommers

Sergey Polzunov

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

Google Inc.
Mark Risher

Hitachi, Ltd.
Kazuo Noguchi
Akihito Sawada
Masato Terada

iboss, Inc.
Paul Martini

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

Intel Corporation
Tim Casey
Kent Landfield

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

LookingGlass
Allan Thomson
Lee Vorthman

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

Rutger Prins
Andrei Sirghi
Raymon van der Velde

eSentire, Inc.
Jacob Gajek

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

Fox-IT
Sarah Brown

Georgetown University
Eric Burger

Hewlett Packard Enterprise (HPE)
Tomas Sander

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

IID
Chris Richardson

Integrated Networking Technologies, Inc.
Patrick Maroney

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

Kaiser Permanente
Russell Culpepper
Beth Pumo

Lumeta Corporation
Brandon Hoffman

MTG Management Consultants, LLC.

Charles Schmidt
Emmanuelle Vargas-Gonzalez
John Wunder

National Council of ISACs (NCI)

Scott Algeier
Denise Anderson
Josh Poster

NEC Corporation

Takahiro Kakumaru

North American Energy Standards Board

David Darnell

Object Management Group

Cory Casanave

Palo Alto Networks

Vishaal Hariprasad

Queralt, Inc.

John Tolbert

Resilient Systems, Inc.

Ted Julian

Securonix

Igor Baikalov

Siemens AG

Bernd Grobauer

Soltra

John Anderson
Aishwarya Asok Kumar
Peter Ayasse
Jeff Beekman
Michael Butt
Cynthia Camacho
Aharon Chernin
Mark Clancy
Brady Cotton
Trey Darley
Mark Davidson
Paul Dion
Daniel Dye
Robert Hutto
Raymond Keckler
Ali Khan
Chris Kiehl

James Cabral

National Security Agency

Mike Boyle
Jessica Fitzgerald-McKay

New Context Services, Inc.

John-Mark Gurney
Christian Hunt
James Moler
Daniel Riedel
Andrew Storms

OASIS

James Bryce Clark
Robin Cover
Chet Ensign

Open Identity Exchange

Don Thibeau

PhishMe Inc.

Josh Larkins

Raytheon Company-SAS

Daniel Wyschogrod

Retail Cyber Intelligence Sharing Center (R-CISC)

Brian Engle

Semper Fortis Solutions

Joseph Brand

Splunk Inc.

Cedric LeRoux
Brian Luger
Kathy Wang

TELUS

Greg Reaume
Alan Steer

Threat Intelligence Pty Ltd

Tyron Miller
Andrew van der Stock

ThreatConnect, Inc.

Wade Baker
Cole Iliff
Andrew Pendergast
Ben Schmoker
Jason Spies

TruSTAR Technology

Clayton Long
Michael Pepin
Natalie Suarez
David Waters
Benjamin Yates

Symantec Corp.

Curtis Kostrosky

The Boeing Company

Crystal Hayes

ThreatQuotient, Inc.

Ryan Trost

U.S. Bank

Mark Angel

Brad Butts

Brian Fay

Mona Magathan

Yevgen Sautin

US Department of Defense (DoD)

James Bohling

Eoghan Casey

Gary Katz

Jeffrey Mates

VeriSign

Robert Coderre

Kyle Maxwell

Eric Osterweil

Chris Roblee

United Kingdom Cabinet Office

Iain Brown

Adam Cooper

Mike McLellan

Chris O'Brien

James Penman

Howard Staple

Chris Taylor

Laurie Thomson

Alastair Treharne

Julian White

Bethany Yates

US Department of Homeland Security

Evette Maynard-Noel

Justin Stekervetz

ViaSat, Inc.

Lee Chieffalo

Wilson Figueroa

Andrew May

Yaana Technologies, LLC

Anthony Rutkowski

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

Appendix B. Revision History

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

¹ To save space, the fully qualified names are not used in this diagram.

² This class is not used in this data model. It is used in the Network Flow data.