CybOXTM Version 2.1.1 Part 38: Network Packet Object

Working Draft 01

08 December 2015

Technical Committee:

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

Chair:

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

Editors:

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

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

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

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

Additional artifacts:

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

* *CybOXTM Version 2.1.1 Part 1: Overview*. [URI]
* *CybOXTM Version 2.1.1 Part 2: Common*. [URI]
* *CybOXTM Version 2.1.1 Part 3: Core*. [URI]
* *CybOXTM Version 2.1.1 Part 4: Default Extensions*. [URI]
* *CybOXTM Version 2.1.1 Part 5: Vocabularies*. [URI]
* *CybOXTM Version 2.1.1 Part 6: UML Model*. [URI]
* *CybOXTM Version 2.1.1 Part 7: API Object*. [URI]
* *CybOXTM Version 2.1.1 Part 8: ARP Cache Object*. [URI]
* *CybOXTM Version 2.1.1 Part 9: AS Object*. [URI]
* *CybOXTM Version 2.1.1 Part 10: Account Object*. [URI]
* *CybOXTM Version 2.1.1 Part 11: Address Object*. [URI]
* *CybOXTM Version 2.1.1 Part 12: Archive File Object*. [URI]
* *CybOXTM Version 2.1.1 Part 13: Artifact Object*. [URI]
* *CybOXTM Version 2.1.1 Part 14: Code Object*. [URI]
* *CybOXTM Version 2.1.1 Part 15: Custom Object*. [URI]
* *CybOXTM Version 2.1.1 Part 16: DNS Cache Object*. [URI]
* *CybOXTM Version 2.1.1 Part 17: DNS Query Object*. [URI]
* *CybOXTM Version 2.1.1 Part 18: DNS Record Object*. [URI]
* *CybOXTM Version 2.1.1 Part 19: Device Object*. [URI]
* *CybOXTM Version 2.1.1 Part 20: Disk Object*. [URI]
* *CybOXTM Version 2.1.1 Part 21: Disk Partition Object*. [URI]
* *CybOXTM Version 2.1.1 Part 22: Domain Name Object*. [URI]
* *CybOXTM Version 2.1.1 Part 23: Email Message Object*. [URI]
* *CybOXTM Version 2.1.1 Part 24: File Object*. [URI]
* *CybOXTM Version 2.1.1 Part 25: GUI Dialogbox Object*. [URI]
* *CybOXTM Version 2.1.1 Part 26: GUI Object*. [URI]
* *CybOXTM Version 2.1.1 Part 27: GUI Window Object*. [URI]
* *CybOXTM Version 2.1.1 Part 28: HTTP Session Object*. [URI]
* *CybOXTM Version 2.1.1 Part 29: Hostname Session Object*. [URI]
* *CybOXTM Version 2.1.1 Part 30: Image File Object*. [URI]
* *CybOXTM Version 2.1.1 Part 31: Library File Object*. [URI]
* *CybOXTM Version 2.1.1 Part 32: Link Object*. [URI]
* *CybOXTM Version 2.1.1 Part 33: Linux Package Object*. [URI]
* *CybOXTM Version 2.1.1 Part 34: Memory Object*. [URI]
* *CybOXTM Version 2.1.1 Part 35: Mutex Object*. [URI]
* *CybOXTM Version 2.1.1 Part 36: Network Connection Object*. [URI]
* *CybOXTM Version 2.1.1 Part 37: Network Flow Object*. [URI]
* *CybOXTM Version 2.1.1 Part 38: Network Packet Object*. (this document)
* *CybOXTM Version 2.1.1 Part 39: Network Route Entry Object*. [URI]
* *CybOXTM Version 2.1.1 Part 40: Network Route Object*. [URI]
* *CybOXTM Version 2.1.1 Part 41: Network Socket Object*. [URI]
* *CybOXTM Version 2.1.1 Part 42: Network Subnet Object*. [URI]
* *CybOXTM Version 2.1.1 Part 43: PDF File Object*. [URI]
* *CybOXTM Version 2.1.1 Part 44: Pipe Object*. [URI]
* *CybOXTM Version 2.1.1 Part 45: Port Object*. [URI]
* *CybOXTM Version 2.1.1 Part 46: Process Object*. [URI]
* *CybOXTM Version 2.1.1 Part 47: Product Object*. [URI]
* *CybOXTM Version 2.1.1 Part 48: SMS Message Object*. [URI]
* *CybOXTM Version 2.1.1 Part 49: Semaphore Object*. [URI]
* *CybOXTM Version 2.1.1 Part 50: Socket Address Object*. [URI]
* *CybOXTM Version 2.1.1 Part 51: System Object*. [URI]
* *CybOXTM Version 2.1.1 Part 52: URI Object*. [URI]
* *CybOXTM Version 2.1.1 Part 53: URL History Object*. [URI]
* *CybOXTM Version 2.1.1 Part 54: Unix File Object*. [URI]
* *CybOXTM Version 2.1.1 Part 55: Unix Network Route Entry Object*. [URI]
* *CybOXTM Version 2.1.1 Part 56: Unix Pipe Object*. [URI]
* *CybOXTM Version 2.1.1 Part 57: Unix Process Object*. [URI]
* *CybOXTM Version 2.1.1 Part 58: Unix User Account Object*. [URI]
* *CybOXTM Version 2.1.1 Part 59: Unix Volume Object*. [URI]
* *CybOXTM Version 2.1.1 Part 60: User Account Object*. [URI]
* *CybOXTM Version 2.1.1 Part 61: User Session Object*. [URI]
* *CybOXTM Version 2.1.1 Part 62: Volume Object*. [URI]
* *CybOXTM Version 2.1.1 Part 63: Whois Object*. [URI]
* *CybOXTM Version 2.1.1 Part 64: Win Computer Account Object*. [URI]
* *CybOXTM Version 2.1.1 Part 65: Win Critical Section Object*. [URI]
* *CybOXTM Version 2.1.1 Part 66: Win Driver Object*. [URI]
* *CybOXTM Version 2.1.1 Part 67: Win Event Log Object*. [URI]
* *CybOXTM Version 2.1.1 Part 68: Win Event Object*. [URI]
* *CybOXTM Version 2.1.1 Part 69: Win Executable File Object*. [URI]
* *CybOXTM Version 2.1.1 Part 70: Win File Object*. [URI]
* *CybOXTM Version 2.1.1 Part 71: Win Filemapping Object*. [URI]
* *CybOXTM Version 2.1.1 Part 72: Win Handle Object*. [URI]
* *CybOXTM Version 2.1.1 Part 73: Win Hook Object*. [URI]
* *CybOXTM Version 2.1.1 Part 74: Win Kernel Hook Object*. [URI]
* *CybOXTM Version 2.1.1 Part 75: Win Kernel Object*. [URI]
* *CybOXTM Version 2.1.1 Part 76: Win Mailslot Object*. [URI]
* *CybOXTM Version 2.1.1 Part 77: Win Memory Page Region Object*. [URI]
* *CybOXTM Version 2.1.1 Part 78: Win Mutex Object*. [URI]
* *CybOXTM Version 2.1.1 Part 79: Win Network Route Entry Object*. [URI]
* *CybOXTM Version 2.1.1 Part 80: Win Network Share Object*. [URI]
* *CybOXTM Version 2.1.1 Part 81: Win Pipe Object*. [URI]
* *CybOXTM Version 2.1.1 Part 82: Win Prefetch Object*. [URI]
* *CybOXTM Version 2.1.1 Part 83: Win Process Object*. [URI]
* *CybOXTM Version 2.1.1 Part 84: Win Registry Key Object*. [URI]
* *CybOXTM Version 2.1.1 Part 85: Win Semaphore Object*. [URI]
* *CybOXTM Version 2.1.1 Part 86: Win Service Object*. [URI]
* *CybOXTM Version 2.1.1 Part 87: Win System Object*. [URI]
* *CybOXTM Version 2.1.1 Part 88: Win System Restore Object*. [URI]
* *CybOXTM Version 2.1.1 Part 89: Win Task Object*. [URI]
* *CybOXTM Version 2.1.1 Part 90: Win Thread Object*. [URI]
* *CybOXTM Version 2.1.1 Part 91: Win User Account Object*. [URI]
* *CybOXTM Version 2.1.1 Part 92: Win Volume Object*. [URI]
* *CybOXTM Version 2.1.1 Part 93: Win Waitable Timer Object*. [URI]
* *CybOXTM Version 2.1.1 Part 94: X509 Certificate Object*. [URI]

Related work:

This specification is related to:

* *STIXTM Version 1.2.1 (placeholder)*

Abstract:

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

Status:

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

URI patterns:

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

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

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

Copyright © OASIS Open 2015. All Rights Reserved.

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full [Policy](https://www.oasis-open.org/policies-guidelines/ipr) may be found at the OASIS website.

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

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

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

Portions copyright © United States Government 2012-2015.  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 8](#_Toc437351544)

[1.1 CybOXTM Specification Documents 8](#_Toc437351545)

[1.2 Document Conventions 8](#_Toc437351546)

[1.2.1 Fonts 8](#_Toc437351547)

[1.2.2 UML Package References 9](#_Toc437351548)

[1.2.3 UML Diagrams 9](#_Toc437351549)

[1.2.3.1 Class Properties 9](#_Toc437351550)

[1.2.3.2 Diagram Icons and Arrow Types 9](#_Toc437351551)

[1.2.4 Property Table Notation 10](#_Toc437351552)

[1.2.5 Property and Class Descriptions 10](#_Toc437351553)

[1.3 Terminology 11](#_Toc437351554)

[1.4 Normative References 11](#_Toc437351555)

[2 Background Information 12](#_Toc437351556)

[2.1 Cyber Observables 12](#_Toc437351557)

[2.2 Objects 12](#_Toc437351558)

[3 Data Model 13](#_Toc437351559)

[3.1 NetworkPacketObjectType Class 13](#_Toc437351560)

[3.2 LinkLayerType Class 14](#_Toc437351561)

[3.3 PhysicalInterfaceType Class 14](#_Toc437351562)

[3.4 LogicalProtocolType Class 15](#_Toc437351563)

[3.5 EthernetInterfaceType Class 15](#_Toc437351564)

[3.6 EthernetHeaderType Class 15](#_Toc437351565)

[3.7 TypeLengthType Class 16](#_Toc437351566)

[3.8 ARPType Class 17](#_Toc437351567)

[3.9 ARPOpType Class 18](#_Toc437351568)

[3.10 NDPType Class 18](#_Toc437351569)

[3.11 RouterSolicitationType Class 19](#_Toc437351570)

[3.12 RouterSolicitationOptionsType Class 19](#_Toc437351571)

[3.13 RouterAdvertisementType Class 20](#_Toc437351572)

[3.14 RouterAdvertisementOptionsType Class 21](#_Toc437351573)

[3.15 NeighborSolicitationType Class 22](#_Toc437351574)

[3.16 NeighborSolicitationOptionsType Class 22](#_Toc437351575)

[3.17 NeighborAdvertisementType Class 22](#_Toc437351576)

[3.18 NeighborOptionsType Class 23](#_Toc437351577)

[3.19 RedirectType Class 24](#_Toc437351578)

[3.20 RedirectOptionsType Class 24](#_Toc437351579)

[3.21 NDPLinkAddrType Class 25](#_Toc437351580)

[3.22 NDPPrefixInfoType Class 25](#_Toc437351581)

[3.23 NDPRedirectedHeaderType Class 26](#_Toc437351582)

[3.24 NDPMTUType Class 27](#_Toc437351583)

[3.25 InternetLayerType Class 27](#_Toc437351584)

[3.26 IPv4PacketType Class 28](#_Toc437351585)

[3.27 IPv4HeaderType Class 29](#_Toc437351586)

[3.28 IPv4FlagsType Class 30](#_Toc437351587)

[3.29 DoNotFragmentType Class 31](#_Toc437351588)

[3.30 MoreFragmentsType Class 31](#_Toc437351589)

[3.31 IPv4OptionType Class 31](#_Toc437351590)

[3.32 IPv4CopyFlagType Class 32](#_Toc437351591)

[3.33 IPv4ClassType Class 32](#_Toc437351592)

[3.34 IPv4OptionsType Class 32](#_Toc437351593)

[3.35 IPv6PacketType Class 32](#_Toc437351594)

[3.36 IPv6HeaderType Class 33](#_Toc437351595)

[3.37 IPv6ExtHeaderType Class 34](#_Toc437351596)

[3.38 IPv6DoNotRecogActionType Class 35](#_Toc437351597)

[3.39 IPv6PacketChangeType Class 35](#_Toc437351598)

[3.40 IPv6OptionType Class 35](#_Toc437351599)

[3.41 IPVersionType Class 36](#_Toc437351600)

[3.42 TransportLayerType Class 36](#_Toc437351601)

[3.43 TCPType Class 37](#_Toc437351602)

[3.44 UDPType Class 37](#_Toc437351603)

[3.45 TCPHeaderType Class 38](#_Toc437351604)

[3.46 TCPFlagsType Class 39](#_Toc437351605)

[3.47 UDPHeaderType Class 40](#_Toc437351606)

[3.48 IANAHardwareType Class 41](#_Toc437351607)

[3.49 IANAEtherType Class 41](#_Toc437351608)

[3.50 IANAAssignedIPNumbersType Class 41](#_Toc437351609)

[3.51 IANAPortNumberRegistryType Class 41](#_Toc437351610)

[3.52 ICMPv4PacketType Class 41](#_Toc437351611)

[3.53 ICMPv4HeaderType Class 42](#_Toc437351612)

[3.54 ICMPv4ErrorMessageType Class 42](#_Toc437351613)

[3.55 ICMPv4ErrorMessageContentType Class 44](#_Toc437351614)

[3.56 ICMPv4InfoMessageType Class 44](#_Toc437351615)

[3.57 ICMPv4InfoMessageContentType Class 45](#_Toc437351616)

[3.58 ICMPv4TracerouteType Class 46](#_Toc437351617)

[3.59 ICMPv6PacketType Class 47](#_Toc437351618)

[3.60 ICMPv6HeaderType Class 48](#_Toc437351619)

[3.61 ICMPv6ErrorMessageType Class 48](#_Toc437351620)

[3.62 ICMPv6InfoMessageType Class 49](#_Toc437351621)

[3.63 ICMPv6InfoMessageContentType Class 50](#_Toc437351622)

[3.64 ICMPv4EchoReplyType Class 50](#_Toc437351623)

[3.65 ICMPv4DestinationUnreachableType Class 51](#_Toc437351624)

[3.66 FragmentationRequiredType Class 53](#_Toc437351625)

[3.67 ICMPv4SourceQuenchType Class 54](#_Toc437351626)

[3.68 ICMPv4RedirectMessageType Class 54](#_Toc437351627)

[3.69 ICMPv4EchoRequestType Class 55](#_Toc437351628)

[3.70 ICMPv4TimeExceededType Class 55](#_Toc437351629)

[3.71 ICMPv4TimestampRequestType Class 56](#_Toc437351630)

[3.72 ICMPv4TimestampReplyType Class 56](#_Toc437351631)

[3.73 ICMPv4AddressMaskRequestType Class 57](#_Toc437351632)

[3.74 ICMPv4AddressMaskReplyType Class 58](#_Toc437351633)

[3.75 ICMPv6DestinationUnreachableType Class 58](#_Toc437351634)

[3.76 ICMPv6PacketTooBigType Class 59](#_Toc437351635)

[3.77 ICMPv6TimeExceededType Class 59](#_Toc437351636)

[3.78 ICMPv6ParameterProblemType Class 60](#_Toc437351637)

[3.79 ICMPv6EchoRequestType Class 61](#_Toc437351638)

[3.80 ICMPv6EchoReplyType Class 61](#_Toc437351639)

[3.81 PrefixType Class 61](#_Toc437351640)

[3.82 HopByHopOptionsType Class 62](#_Toc437351641)

[3.83 OptionDataType Class 62](#_Toc437351642)

[3.84 RoutingType Class 63](#_Toc437351643)

[3.85 FragmentType Class 64](#_Toc437351644)

[3.86 DestinationOptionsType Class 64](#_Toc437351645)

[3.87 AuthenticationHeaderType Class 65](#_Toc437351646)

[3.88 EncapsulatingSecurityPayloadType Class 66](#_Toc437351647)

[3.89 Pad1Type Class 67](#_Toc437351648)

[3.90 PadNType Class 67](#_Toc437351649)

[3.91 FragmentHeaderType Class 68](#_Toc437351650)

[3.92 MFlagType Class 69](#_Toc437351651)

[3.93 ARPOpTypeEnum Enumeration 69](#_Toc437351652)

[3.94 DoNotFragmentTypeEnum Enumeration 69](#_Toc437351653)

[3.95 MoreFragmentsTypeEnum Enumeration 70](#_Toc437351654)

[3.96 IPv4CopyFlagTypeEnum Enumeration 70](#_Toc437351655)

[3.97 IPv4ClassTypeEnum Enumeration 70](#_Toc437351656)

[3.98 IPv4OptionsTypeEnum Enumeration 71](#_Toc437351657)

[3.99 IPv6DoNotRecogActionTypeEnum Enumeration 72](#_Toc437351658)

[3.100 IPv6PacketChangeTypeEnum Enumeration 73](#_Toc437351659)

[3.101 IPVersionTypeEnum Enumeration 73](#_Toc437351660)

[3.102 IANAHardwareTypeEnum Enumeration 74](#_Toc437351661)

[3.103 IANAEtherTypeEnum Enumeration 75](#_Toc437351662)

[3.104 IANAAssignedIPNumbersTypeEnum Enumeration 75](#_Toc437351663)

[3.105 IANAPortNumberRegistryTypeEnum Enumeration 77](#_Toc437351664)

[3.106 MFlagTypeEnum Enumeration 78](#_Toc437351665)

[4 Conformance 79](#_Toc437351666)

[Acknowledgments 80](#_Toc437351667)

[Revision History 81](#_Toc437351668)

# Introduction

[All text is normative unless otherwise labeled]

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

This document serves as the specification for the CybOX Network 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 in, 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**.

## CybOXTM Specification Documents

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

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

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

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

## Document Conventions

The following conventions are used in this document.

### Fonts

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

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

Examples: Action, Object, Event, Property

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

Examples: ActionType, cyboxCommon:BaseObjectPropertyType

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

* The ‘*italic’* font (withsingle quotes) is used for noting actual, explicit values for CybOX Language properties. The *italic* font (without quotes) is used for noting example values.

Example:  *‘HashNameVocab-1.0,’ high, medium, low*

### UML Package References

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

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.

### UML Diagrams

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

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

#### Class Properties

Generally, a class property can be shown in a UML diagram as either an attribute or an association (i.e., the distinction between attributes and associations is somewhat subjective). In order to make the size of UML diagrams in the specifications manageable, we have chosen to capture most properties as attributes and to capture only higher level properties as associations, especially in the main top-level component diagrams. In particular, we will always capture properties of UML data types as attributes. For example, properties of a class that are identifiers, titles, and timestamps will be represented as attributes.

#### Diagram Icons and Arrow Types

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

Table ‑. UML diagram icons

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

### Property Table Notation

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

### Property and Class Descriptions

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

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

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

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

## Terminology

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

## Normative References

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

# Background Information

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

## Cyber Observables

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

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

## Objects

Objects in CybOX are individual data models for characterizing 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.

# Data Model

## NetworkPacketObjectType Class

The NetworkPacketObjectType class’ definition of a network packet is based on the TCP/IP model/Internet protocol suite. In the TCP/IP stack, "packet" is generally defined as IP header plus payload, but we also include the LinkLayer from the OSI model, which defines the physical network interfaces and routing protocols. Protocol fields are provided but requirements are not enforced/captured; all fields are optional. The UML diagram corresponding to the NetworkPacketObjectType class is shown in **Figure 3‑1**.

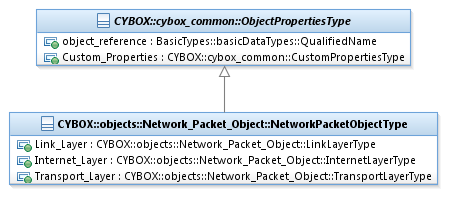


Figure ‑. UML diagram of the NetworkPacketObjectType class

The property table of the NetworkPacketObjectType class is given in **Table 3‑1**.

Table ‑. Properties of the NetworkPacketObjectType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Link\_Layer** | PacketObj:  LinkLayerType | 0..1 | The Link\_Layer 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. |
| **Internet\_Layer** | PacketObj:  InternetLayerType | 0..1 | The Internet\_Layer property characterizes information about the network layer of this Network Packet. The network layer is one layer from the 7-layer OSI Model. |
| **Transport\_Layer** | PacketObj:  TransportLayerType | 0..1 | The Transport\_Layer property characterizes information about the transport layer of this Network Packet. The transport layer is one layer from the 7-layer OSI Model. |

## LinkLayerType Class

A link layer protocol is a hardware interface protocol, such as Ethernet, or a logical link routing protocol, such as ARP.

The property table of the LinkLayerType class is given in **Table 3‑2**.

Table ‑. Properties of the LinkLayerType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Physical\_Interface** | PacketObj:  PhysicalInterfaceType | 0..1 | The Physical\_Interface property characterizes one hardware interface of a link layer connection. |
| **Logical\_Protocols** | PacketObj:  LogicalProtocolType | 0..1 | The Logical\_Protocols property characterizes the logical protocol of a link layer connection. One example of a logical protocol is ARP. |

## PhysicalInterfaceType Class

Multiple interface classs exist - only most common (Ethernet) included now. Others will be added later as needed.

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

Table ‑. Properties of the PhysicalInterfaceType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Ethernet** | PacketObj:  EthernetInterfaceType | 0..1 | The Ethernet property sends network packets from the sending host to one or more receiving hosts. (REF: IEEE 802.3; http://wiki.wireshark.org/Ethernet). |

## LogicalProtocolType Class

Logical Protocols characterizes the logical protocol of a link layer connection. One example of a logical protocol is ARP.

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

Table ‑. Properties of the LogicalProtocolType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **ARP\_RARP** | PacketObj:ARPType | 0..1 | The ARP\_RARP property 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. |
| **NDP** | PacketObj:NDPType | 0..1 | The NDP property (Neighbor Discovery Protocol) is used with IPv6 to determine the link-layer addresses for neighbors. Corresponds to combination of IPv4 protocols: ARP, ICMP Router Discovery, and ICMP Redirect. |

## EthernetInterfaceType Class

Ethernet sends network packets from the sending host to one or more receiving hosts. (REF: IEEE 802.3; http://wiki.wireshark.org/Ethernet).

The property table of the EthernetInterfaceType class is given in **Table 3‑5**.

Table ‑. Properties of the EthernetInterfaceType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Ethernet\_Header** | PacketObj:  EthernetHeaderType | 0..1 | The Ethernet\_Header property includes information such as source MAC address, destination MAC address, and more. |

## EthernetHeaderType Class

Ethernet header characterizes and 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 3‑6**.

Table ‑. Properties of the EthernetHeaderType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Destination\_MAC\_Addr** | AddressObj:  AddressObjectType | 0..1 | The Destination\_MAC\_Addr property characterizes the destination MAC Address of the ethernet frame. |
| **Source\_MAC\_Addr** | AddressObj:  AddressObjectType | 0..1 | The Source\_MAC\_Addr property characterizes the source MAC Address of the ethernet frame. |
| **Type\_Or\_Length** | PacketObj:  TypeLengthType | 0..1 | The Type\_Or\_Length property characterizes either the length of the ethernet frame or the protocol type of the network layer. |
| **Checksum** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Checksum property characterizes the Frame Check sequence of an ethernet frame. |

## TypeLengthType Class

0-1500 then it is a length field. Otherwise, it defines the protocol class of the Internet layer.

The property table of the TypeLengthType class is given in **Table 3‑7**.

Table ‑. Properties of the TypeLengthType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Length** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Length property haracterizes the length of the ethernet frame. |
| **Internet\_Layer\_Type** | PacketObj:IANAEtherType | 0..1 | two-octet property in an Ethernet frame. Specifies protocol encapsulated in the payload of ethernet frame. |

## ARPType Class

The Address Resolution Protocol is a request and reply protocol that runs encapsulated by the line protocol. It is communicated within the boundaries of a single network, never routed across internetwork nodes. This property places ARP into the Link Layer. It is encapsulated. REF: http://www.comptechdoc.org/independent/networking/guide/netarp.html.

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

Table ‑. Properties of the ARPType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Hardware\_Addr\_Type** | PacketObj:IANAHardwareType | 0..1 | The Hardware\_Addr\_Type property characterizes the type of hardware address specified in an ARP message. |
| **Proto\_Addr\_Type** | PacketObj:IANAEtherType | 0..1 | The Proto\_Addr\_Type property characterizes the type of protocol address being mapped. For IPv4 addresses, value = 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. IPv4 addresses = 4. |
| **Op\_Type** | PacketObj:ARPOpType | 0..1 | The Op\_Type property characterizes the type of operation. 1 = ARP request, 2=ARP reply, 3=RARP request, 4=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:  AddressObjectType | 0..1 | The Sender\_Protocol\_Addr property characterizes the 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. |

## ARPOpType Class

The ARPOpType class specifies class of ARP operations, via a union of the ARPOpTypeEnum enumeration and the atomic xs:string type. Its base type is the BaseObjectPropertyType class, for permitting complex (i.e. regular-expression based) specifications.

## NDPType Class

NDP Type characterizes NDP (Neighbor Discover Protocol) IPv6 packets. NDP defines five ICMPv6 packet classs. RFC 2461: http://tools.ietf.org/html/rfc4861.

The property table of the NDPType class is given in **Table 3‑9**.

Table ‑. Properties of the NDPType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **ICMPv6\_Header** | PacketObj:  ICMPv6HeaderType | 0..1 | The ICMPv6\_Header property characterizes an ICMPv6 header. |
| **Router\_Solicitation** | PacketObj:  RouterSolicitationType | 0..1 | The Router\_Solicitation property - Hosts send Router Solicitations in order to prompt routers to generate Router Advertisements quickly (type=133; code=0). |
| **Router\_Advertisement** | PacketObj:  RouterAdvertisementType | 0..1 | The Router\_Advertisement property - Routers send out Router Advertisement messages periodically, or in response to Router Solicitations (type=134; code=0). |
| **Neighbor\_Solicitation** | PacketObj:  NeighborSolicitationType | 0..1 | The Neighbor\_Solicitation property - 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). |
| **Neighbor\_Advertisement** | PacketObj:  NeighborAdvertisementType | 0..1 | The Neighbor\_Advertisement property - 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). |
| **Redirect** | PacketObj:RedirectType | 0..1 | The Redirect property - 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). |

## RouterSolicitationType Class

Hosts send Router Solicitations 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 ‑. Properties of the RouterSolicitationType class

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

## RouterSolicitationOptionsType Class

Neighbor Discovery messages include zero or more options, 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 ‑. Properties of the RouterSolicitationOptionsType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Src\_Link\_Addr** | PacketObj:NDPLinkAddrType | 0..1 | The Src\_Link\_Addr property characterizes the Source Link-Layer Address option. |

## RouterAdvertisementType Class

Routers send out Router Advertisement messages 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 ‑. Properties of the RouterAdvertisementType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **managed\_address\_**  **config\_flag** | basicDataTypes:Boolean | 0..1 | The managed\_address\_config\_flag property 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 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 | 8-bit unsigned integer. 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 | 16-bit unsigned integer. 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 Reachable\_Time property 32-bit unsigned integer. 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 Retrans\_Timer property 32-bit unsigned integer. 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** | PacketObj:  RouterAdvertisementOptionsType | 0..1 | The Options property Neighbor Discovery messages include zero or more options, some of which may appear multiple times in the same message. |

## RouterAdvertisementOptionsType Class

Router Advertisement messages include zero or more options, 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 ‑. Properties of the RouterAdvertisementOptionsType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Src\_Link\_Addr** | PacketObj:NDPLinkAddrType | 0..1 | The Src\_Link\_Addr property Src Link Addr characterizes the Source Link-Layer Address option. |
| **MTU** | PacketObj:NDPMTUType | 0..1 | The MTU property 32-bit unsigned integer. The recommended MTU for the link. |
| **Prefix\_Info** | PacketObj:NDPPrefixInfoType | 0..1 | The Prefix\_Info property Prefix Info characterizes Prefix Information for Router Advertisement Options. |

## NeighborSolicitationType 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. (class=135; code=0).

The property table of the NeighborSolicitationType class is given in **Table 3‑14**.

Table ‑. Properties of the NeighborSolicitationType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Target\_IPv6\_Addr** | AddressObj:AddressObjectType | 0..1 | The Target\_IPv6\_Addr property The IP address of the target of the solicitation. |
| **Options** | PacketObj:  NeighborSolicitationOptionsType | 0..1 | The Options property Neighbor Solicitation messages include zero or more options, some of which may appear multiple times in the same message. |

## NeighborSolicitationOptionsType Class

Neighbor Solicitation messages include zero or more options, some of which may appear multiple times in the same message.

The property table of the NeighborSolicitationOptionsType class is given in **Table 3‑15**.

Table ‑. Properties of the NeighborSolicitationOptionsType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Src\_Link\_Addr** | PacketObj:NDPLinkAddrType | 0..1 | The Src\_Link\_Addr property Src Link Addr characterizes the Source Link-Layer Address option. |

## NeighborAdvertisementType Class

A node sends Neighbor Advertisements 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 3‑16**.

Table ‑. Properties of the NeighborAdvertisementType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **router\_flag** | basicDataTypes:  Boolean | 0..1 | The router\_flag property Router flag. 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 solicited\_flag property Solicited flag. 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 override\_flag property Override flag. 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 Target\_IPv6\_Addr property The IP address of the target of the advertisement. |
| **Options** | PacketObj:  NeighborOptionsType | 0..1 | The Options property Neighbor Advertisement messages include zero or more options, some of which may appear multiple times in the same message. |

## NeighborOptionsType Class

Neighbor Advertisement messages include zero or more options, some of which may appear multiple times in the same message.

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

Table ‑. Properties of the NeighborOptionsType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Target\_Link\_Addr** | PacketObj:NDPLinkAddrType | 0..1 | The Target\_Link\_Addr property Target Link Addr characterizes the Target Link-Layer Address option. |

## RedirectType 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. (class=137; code=0).

The property table of the RedirectType class is given in **Table 3‑18**.

Table ‑. Properties of the RedirectType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Target\_IPv6\_Addr** | AddressObj:  AddressObjectType | 0..1 | The Target\_IPv6\_Addr property An IP address that is a better first hop to use for the ICMP Destination Address. |
| **Dest\_IPv6\_Addr** | AddressObj:  AddressObjectType | 0..1 | The Dest\_IPv6\_Addr property The IP address of the destination that is redirected to the target. |
| **Options** | PacketObj:  RedirectOptionsType | 0..1 | The Options property Redirect messages include zero or more options, some of which may appear multiple times in the same message. |

## RedirectOptionsType Class

Redirect messages include zero or more options, some of which may appear multiple times in the same message.

The property table of the RedirectOptionsType class is given in **Table 3‑19**.

Table ‑. Properties of the RedirectOptionsType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Target\_Link\_Addr** | PacketObj:  NDPLinkAddrType | 0..1 | The Target\_Link\_Addr property The link-layer address for the target. |
| **Redirected\_Header** | PacketObj:  NDPRedirectedHeaderType | 0..1 | The Redirected\_Header property As much as possible of the IP packet that triggered the sending of the Redirect message without making the redirect packet exceed the minimum MTU specified in the IPv6 protocol. |

## NDPLinkAddrType Class

NDPLinkAddrType characterizes the Link-Layer Address option.

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

Table ‑. Properties of the NDPLinkAddrType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Length** | cyboxCommon:  IntegerObjectPropertyType | 0..1 | The length of the option (including the type and length propertys) in units of 8 octets. |
| **Link\_Layer\_MAC\_Addr** | AddressObj:  AddressObjectType | 0..1 | The 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. |

## NDPPrefixInfoType Class

Prefix Info characterizes Prefix Information for Router Advertisement Options. It provides hosts with on-link prefixes and prefixes for Address Autoconfiguration. (class=3). RFC 4861.

The property table of the NDPPrefixInfoType class is given in **Table 3‑21**.

Table ‑. Properties of the NDPPrefixInfoType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **link\_flag** | basicDataTypes:Boolean | 0..1 | The link\_flag property 1-bit on-link flag. When set, indicates that this prefix can be used for on-link determintation. When not set the advertisement makes no statement about on-link or off-link properties of the prefix. |
| **addr\_config\_flag** | basicDataTypes:Boolean | 0..1 | The addr\_config\_flag property 1-bit autonomous address-configuration flag. When set indicates that this prefix can be usd for stateless address configuration. |
| **Length** | cyboxCommon:  IntegerObjectPropertyType | 0..1 | The Length property Length 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** | cyboxCommon:  IntegerObjectPropertyType | 0..1 | 8-bit unsigned integer. 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** | cyboxCommon:  IntegerObjectPropertyType | 0..1 | The Valid\_Lifetime property 32-bit unsigned integer. 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 32-bit unsigned integer. 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** | PacketObj:PrefixType | 0..1 | The Prefix property The Prefix is an IP address or a prefix of an IP address. |

## NDPRedirectedHeaderType Class

The redirected header option is used in redirect messages 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 3‑22**.

Table ‑. Properties of the NDPRedirectedHeaderType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Length** | cyboxCommon:  IntegerObjectPropertyType | 0..1 | The length of the option (including the type and length propertys) in units of 8 octets. |
| **IPHeader\_And\_Data** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The IPHeader\_And\_Data property As much as possible of the IP packet that triggered the sending of the redirect without making redirect packet larger than MTU. |

## NDPMTUType Class

The MTU option 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 3‑23**.

Table ‑. Properties of the NDPMTUType class

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

## InternetLayerType Class

The Internet layer is the group of methods, protocols, and specifications that are used to transport packets from the originating host across network boundaries. Not all protocols are currently defined, just those most commonly used: IPv4, ICMPv4, IPv6, ICMPv6. Other protocols will be added as needed. (http://en.wikipedia.org/wiki/Internet\_layer).

The property table of the InternetLayerType class is given in **Table 3‑24**.

Table ‑. Properties of the InternetLayerType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **IPv4** | PacketObj:IPv4PacketType | 0..1 | The IPv4 property Internet Protocol version 4 (IPv4) is a connectionless protocol for use on packet-switched link layer networks (e.g., Ethernet). |
| **ICMPv4** | PacketObj:ICMPv4PacketType | 0..1 | The ICMPv4 property ICMP is chiefly used 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 (http://en.wikipedia.org/wiki/Internet\_Control\_Message\_Protocol; REF: http://www.networksorcery.com/enp/protocol/icmp.htm). |
| **IPv6** | PacketObj:IPv6PacketType | 0..1 | The IPv6 property Internet Protocol version 6 (IPv6) is intended to succeed IPv4, and like IPv4 it is a connectionless protocol for use on packet-switched link layer networks. |
| **ICMPv6** | PacketObj:ICMPv6PacketType | 0..1 | The ICMPv6 property ICMPv6 is the implementation of the ICMP for IPv6. ICMPv6 performs error reporting and diagnostic functions. |

## IPv4PacketType Class

Internet Protocol version 4 (IPv4) is a connectionless protocol for use on packet-switched link layer networks (e.g., Ethernet). REF: RFC 791; http://en.wikipedia.org/wiki/IPv4.

The property table of the IPv4PacketType class is given in **Table 3‑25**.

Table ‑. Properties of the IPv4PacketType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **IPv4\_Header** | PacketObj:IPv4HeaderType | 0..1 | The IPv4 header provides addressing, and internet modules use propertys in the header to fragment and reassemble internet datagrams when necessary for transmission through small packet networks. |
| **Data** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The data portion of an IP packet is interpreted based on the value of the Protocol header property. Actual field 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. |

## IPv4HeaderType Class

The IPv4 header provides addressing, and internet modules use fields in the header to fragment and reassemble internet datagrams when necessary for transmission through small packet networks. REF: RFC 791.

The property table of the IPv4HeaderType class is given in **Table 3‑26**.

Table ‑. Properties of the IPv4HeaderType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **IP\_Version** | PacketObj:IPVersionType | 0..1 | The version property indicates the format of the internet header. For IP v4, the version is 4. |
| **Header\_Length** | cyboxCommon:  IntegerObjectPropertyType | 0..1 | The Header\_Length property The Internet Header Length specifies the length of IP packet header in 32 bit words. Min value = 5. |
| **DSCP** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | Originally defined as the Type of Service property, the Differentiated Services Code Point (DSCP) field is now defined by RFC 2474 for Differentiated services (DiffServ). 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 (http://en.wikipedia.org/wiki/IPv4). |
| **ECN** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | Explicit Congestion Notification: This property is defined in RFC 3168 and 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. (http://en.wikipedia.org/wiki/IPv4). |
| **Total\_Length** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | This 16-bit property defines the entire datagram size, including header and data, in bytes. |
| **Identification** | cyboxCommon:  PositiveIntegerObjectPropertyType | 0..1 | The Identification property is primarily used for uniquely identifying fragments of an original IP datagram. (http://en.wikipedia.org/wiki/IPv4). |
| **Flags** | PacketObj:IPv4FlagsType | 0..1 | This is a three-bit property used to control or identify fragments. An field has been defined for each bit with associated enumerated types. |
| **Fragment\_Offset** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The fragment offset property is 13 bits long and specifies the offset of a particular fragment relative to the beginning of the original unfragmented IP datagram. http://en.wikipedia.org/wiki/IPv4. |
| **TTL** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | This 8-bit property helps prevent datagrams from persisting on an internet (it limits a datagram's lifetime). |
| **Protocol** | PacketObj:  IANAAssignedIPNumbersType | 0..1 | This property defines the protocol used in the data portion of the IP datagram. The type of this field is an enumerated list of IP protocol numbers as maintained by the Internet Assigned Numbers Authority. |
| **Checksum** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | This property is a 16-bit checksum used for error-checking of the header. |
| **Src\_IPv4\_Addr** | AddressObj:AddressObjectType | 0..1 | This property is the IPv4 address of the sender of the packet. |
| **Dest\_IPv4\_Addr** | AddressObj:AddressObjectType | 0..1 | This property is the IPv4 address of the receiver of the packet. |
| **Option** | PacketObj:IPv4OptionType | 0..\* | The IPv4 option property is variable in length with zero or more options. It is not often used. http://en.wikipedia.org/wiki/IPv4. |

## IPv4FlagsType Class

These flag classs are used to control or identify fragments in an IP packet. It is a three-bit field, each of the three bits are defined by a field with a string value that indicates the meaning of whether or not the bit is set.

The property table of the IPv4FlagsType class is given in **Table 3‑27**.

Table ‑. Properties of the IPv4FlagsType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Reserved** | cyboxCommon:  IntegerObjectPropertyType | 0..1 | The Reserved property Bit 0: This bit value (0) is reserved and must be zero. |
| **Do\_Not\_Fragment** | PacketObj:DoNotFragmentType | 0..1 | The Do\_Not\_Fragment property Bit 1: This is the "don't fragment" bit. Values are specified in the DoNotFragmentType. |
| **More\_Fragments** | PacketObj:MoreFragmentsType | 0..1 | The More\_Fragments property Bit 2: This is the "more fragments" bit. Values are specified in the MoreFragmentsType. |

## DoNotFragmentType Class

DoNotFragmentType class specifies fragmenting options, via a union of the DoNotFragmentTypeEnum enumeration and the atomic xs:string type. Its base type is the BaseObjectPropertyType class, for permitting complex (i.e. regular-expression based) specifications.

## MoreFragmentsType Class

MoreFragmentsType class specifies whether there are more fragments, via a union of the MoreFragmentsTypeEnum enumeration and the atomic xs:string type. Its base type is the BaseObjectPropertyType class, for permitting complex (i.e. regular-expression based) specifications.

## IPv4OptionType Class

The IPv4 option field is variable in length with zero or more options.

The property table of the IPv4OptionType class is given in **Table 3‑28**.

Table ‑. Properties of the IPv4OptionType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Copy\_Flag** | PacketObj:IPv4CopyFlagType | 0..1 | The copied flag indicates that this option is copied into all fragments on fragmentation. 1 bit. They are represented in this property by a string which specifies their value. |
| **Class** | PacketObj:IPv4ClassType | 0..1 | The option class is represented by 2 bits where 0 = control; 1 = reserved for future use; 2 = debugging and measurement; 3 = reserved for future use. These enumerated values are defined for this property. |
| **Option** | PacketObj:IPv4OptionsType | 0..1 | The Internet Protocol has provision for optional header propertys identified by an option type. These types are enumerated in the IPv4OptionsType class. |

## IPv4CopyFlagType Class

IPv4CopyFlagType class specifies value of IPv4 copy flag, via a union of the IPv4CopyFlagTypeEnum enumeration and the atomic xs:string type. Its base type is the BaseObjectPropertyType class, for permitting complex (i.e. regular-expression based) specifications.

## IPv4ClassType Class

IPv4ClassType class specifies IPv4 class, via a union of the IPv4ClassTypeEnum enumeration and the atomic xs:string type. Its base type is the BaseObjectPropertyType class, for permitting complex (i.e. regular-expression based) specifications.

## IPv4OptionsType Class

IPv4OptionsType class specifies IPv4 options, via a union of the IPv4OptionsTypeEnum enumeration and the atomic xs:string type. Its base type is the BaseObjectPropertyType class, for permitting complex (i.e. regular-expression based) specifications.

## IPv6PacketType Class

Internet Protocol version 6 (IPv6) is intended to succeed IPv4, and like IPv4 it is a connectionless protocol for use on packet-switched link layer networks. RFC 3513, RFC 2460, http://en.wikipedia.org/wiki/IPv6.

The property table of the IPv6PacketType class is given in **Table 3‑29**.

Table ‑. Properties of the IPv6PacketType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **IPv6\_Header** | PacketObj:  IPv6HeaderType | 0..1 | The IPv6\_Header property IPv6 headers is a simplification of the IPv4 header. |
| **Ext\_Headers** | PacketObj:  IPv6ExtHeaderType | 0..\* | The Ext\_Headers property In IPv6, optional internet-layer information is encoded in separate headers that may be placed between the IPv6 header and the upper-layer header in a packet. http://tools.ietf.org/html/rfc2460. |
| **Data** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | 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. |

## IPv6HeaderType Class

The IPv6 header is a simplification of the IPv4 header.

The property table of the IPv6HeaderType class is given in **Table 3‑30**.

Table ‑. Properties of the IPv6HeaderType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **IP\_Version** | PacketObj:IPVersionType | 0..1 | The IP\_Version property 4-bit Internet Protocol version number =6. |
| **Traffic\_Class** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | 8-bit traffic class property. Available for use by originating nodes and/or forwarding routers to identify and distinguish between different classes or priorities of IPv6 packets. http://tools.ietf.org/html/rfc2460#section-7. |
| **Flow\_Label** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Flow\_Label property 20-bit flow label. 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. http://tools.ietf.org/html/rfc2460#section-6. |
| **Payload\_Length** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Payload\_Length property 16-bit unsigned integer. Length of the IPv6 payload (the rest of the packet following the IPv6 header) in octets. Any extension headers are considered part of the payload. |
| **Next\_Header** | PacketObj:  IANAAssignedIPNumbersType | 0..1 | 8-bit selector. Identifies the type of header immediately following the IPv6 header. Uses the same values as the IPv4 protocol property. |
| **TTL** | cyboxCommon:  PositiveIntegerObjectPropertyType | 0..1 | The TTL property TTL/hop limit specifies how many times a packet can be forwarded. 8-bit unsigned integer. |
| **Src\_IPv6\_Addr** | AddressObj:AddressObjectType | 0..1 | The Src\_IPv6\_Addr property 128-bit address of the originator of the packet. |
| **Dest\_IPv6\_Addr** | AddressObj:AddressObjectType | 0..1 | The Dest\_IPv6\_Addr property 128-bit address of the intended recipient of the packet. |

## IPv6ExtHeaderType Class

In IPv6, optional internet-layer information is encoded in separate headers that may be 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 identified by the Next Header field of the preceding header. http://tools.ietf.org/html/rfc2460.

The property table of the IPv6ExtHeaderType class is given in ???.

Table ‑. Properties of the IPv6ExtHeaderType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Hop\_by\_Hop\_**  **Options** | PacketObj:  HopByHopOptionsType | 0..1 | The Hop\_by\_Hop\_Options property The Hop-by-Hop Options header 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. |
| **Routing** | PacketObj:RoutingType | 0..1 | The Routing property The Routing header is used by an IPv6 source to list one or more intermediate nodes to be "visited" on the way to a packet's destination. http://tools.ietf.org/html/rfc2460. |
| **Fragment** | PacketObj:FragmentType | 0..1 | The Fragment header 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 elements. We provide the elements necessary for the Fragmentable Part. http://tools.ietf.org/html/rfc2460. |
| **Destination\_Options** | PacketObj:  DestinationOptionsType | 0..2 | The Destination\_Options property The Destination Options header is used to carry optional information that needs to be examined only by a packet's destination node(s). |
| **Authentication\_**  **Header** | PacketObj:  AuthenticationHeaderType | 0..1 | The Authentication\_Header property Follows RFC2402. The IP Authentication Header is used to provide connectionless integrity and data origin authentication for IP datagrams and to provide protection against replays. http://www.ietf.org/rfc/rfc2402.txt. |
| **Encapsulating\_**  **Security\_Payload** | PacketObj:  EncapsulatingSecurityPayloadType | 0..1 | The Encapsulating\_Security\_Payload property Follows RFC2406. ESP 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. |

## IPv6DoNotRecogActionType Class

IPv6DoNotRecogActionType class specifies possible actions when option is not recognized, via a union of the IPv6DoNotRecogActionTypeEnum enumeration and the atomic xs:string type. Its base type is the BaseObjectPropertyType class, for permitting complex (i.e. regular-expression based) specifications.

## IPv6PacketChangeType Class

IPV6PacketChangeType class specifies whether a packet has changed, via a union of the IPv6PacketChangeTypeEnum enumeration and the atomic xs:string type. Its base type is the BaseObjectPropertyType class, for permitting complex (i.e. regular-expression based) specifications.

## IPv6OptionType Class

Specifies the meaning of each bit of the 8-bit IPv6OptionType class.

The property table of the IPv6OptionType class is given in **Table 3‑32**.

Table ‑. Properties of the IPv6OptionType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Do\_Not\_Recogn\_Action** | PacketObj:  IPv6DoNotRecogActionType | 0..1 | The Do\_Not\_Recogn\_Action property Action to be taken if the processing IPv6 nodes does 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 if the processing IPv6 node does not recognize the Option type. These possible actions are enumerated via IPv6DoNotRecogActionType. |
| **Packet\_Change** | PacketObj:  IPv6PacketChangeType | 0..1 | The Packet\_Change property The third highest order bit of the Option Data specifies whether or not the Option Data of that option can change en-route to the packet's final destination. |
| **Option\_Byte** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | This property may be used to specify the actual Option Type byte, with no explicit meaning attached. Meaning/interpretation provided by the Do\_Not\_Recogn\_Action and Packet\_Change fields. |

## IPVersionType Class

IPVersionType class specifies IP versions, via a union of the IPVersionTypeEnum enumeration and the atomic xs:string type. See http://www.iana.org/assignments/version-numbers/version-numbers.xml for a complete list. Its base type is the BaseObjectPropertyType class, for permitting complex (i.e. regular-expression based) specifications.

## TransportLayerType Class

Only UDP and TCP defined to begin. Other protocols will be defined as necessary.

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

Table ‑. Properties of the TransportLayerType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **TCP** | PacketObj:TCPType | 0..1 | The TCP property TCP provides reliable, ordered delivery of a stream of bytes from a program on one computer to another program on another computer. http://en.wikipedia.org/wiki/Transmission\_Control\_Protocol. |
| **UDP** | PacketObj:UDPType | 0..1 | The UDP property 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. http://en.wikipedia.org/wiki/User\_Datagram\_Protocol. |

## TCPType Class

TCP provides reliable, ordered delivery of a stream of bytes from a program on one computer to another program on another computer. http://en.wikipedia.org/wiki/Transmission\_Control\_Protocol.

The property table of the TCPType class is given in **Table 3‑34**.

Table ‑. Properties of the TCPType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **TCP\_Header** | PacketObj:TCPHeaderType | 0..1 | The TCP header contains 10 mandatory properties and an optional extension field. http://en.wikipedia.org/wiki/Transmission\_Control\_Protocol. |
| **Options** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | Options have up to three propertys: Option-Kind (1 byte), Option-Length (1 byte), Option-Data (variable). This field will be further defined when required. |
| **Data** | cyboxCommon:DataSegmentType | 0..1 | The Data property specifies the data payload of the TCP packet. |

## UDPType Class

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. http://en.wikipedia.org/wiki/User\_Datagram\_Protocol.

The property table of the UDPType class is given in **Table 3‑35**.

Table ‑. Properties of the UDPType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **UDP\_Header** | PacketObj:UDPHeaderType | 0..1 | The UDP header consists of four propertys, which are defined here. |
| **Data** | cyboxCommon:DataSegmentType | 0..1 | The Data property specifies the data payload of the UDP packet. |

## TCPHeaderType Class

The TCP header contains 10 mandatory fields and an optional extension field. http://en.wikipedia.org/wiki/Transmission\_Control\_Protocol.

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

Table ‑. Properties of the TCPHeaderType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Src\_Port** | PortObj:PortObjectType | 0..1 | The Src\_Port property Identifies the sending port. |
| **Dest\_Port** | PortObj:PortObjectType | 0..1 | The Dest\_Port property Identifies the receiving port. |
| **Seq\_Num** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Seq\_Num property The Sequence number (32-bits) has a dual role: If the SYN flag is set, then this is the initial sequence numbers. If the SYN flag is clear (see Control Bits element), then this is the accumulated sequence number of the first data byte of this packet for the current session. http://en.wikipedia.org/wiki/Transmission\_Control\_Protocol. |
| **ACK\_Num** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | If the ACK flag (see Control Bits element) 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 these 3 bits are reserved for future use and should be set to zero. |
| **TCP\_Flags** | PacketObj:TCPFlagsType | 0..1 | The TCP\_Flags property The TCP header contains 9 flags (aka Control Bits). |
| **Window** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | 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. http://en.wikipedia.org/wiki/Transmission\_Control\_Protocol. |
| **Checksum** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The 16-bit checksum property is used for error-checking of the header and data. http://en.wikipedia.org/wiki/Transmission\_Control\_Protocol. |
| **Urg\_Ptr** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | If the URG flag is set, then this 16-bit property is an offset from the sequence number indicating the last urgent data byte. http://en.wikipedia.org/wiki/Transmission\_Control\_Protocol. |

## TCPFlagsType Class

Defines the 9 different flags in the TCP header.

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

Table ‑. Properties of the TCPFlagsType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **ns** | basicDataTypes:Boolean | 0..1 | The ns property ECN-nonce concealment protection. |
| **cwr** | basicDataTypes:Boolean | 0..1 | The cwr property 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. http://en.wikipedia.org/wiki/Transmission\_Control\_Protocol. |
| **ece** | basicDataTypes:Boolean | 0..1 | The ece property ECN-Echo indicates: if the SYN flag is set, that the TCP peer is ECN capable; if the SYN flag is clear, that a packet with Congestion Experienced flag in IP header set is received during normal transmission. http://en.wikipedia.org/wiki/Transmission\_Control\_Protocol. |
| **urg** | basicDataTypes:Boolean | 0..1 | Indicates that the Urgent point property is significant. |
| **ack** | basicDataTypes:Boolean | 0..1 | indicates that the Acknowledgment property is significant. All packets after the initial SYN packet sent by the client should have this flag set. http://en.wikipedia.org/wiki/Transmission\_Control\_Protocol. |
| **psh** | basicDataTypes:Boolean | 0..1 | The psh property Push functions. asks to push the buffered dtata to the receiving application. http://en.wikipedia.org/wiki/Transmission\_Control\_Protocol. |
| **rst** | basicDataTypes:Boolean | 0..1 | The rst property Reset the connection. |
| **syn** | basicDataTypes:Boolean | 0..1 | The syn property Synchronize sequence numbers. Only the first packet sent from each end should have this flag set. http://en.wikipedia.org/wiki/Transmission\_Control\_Protocol. |
| **fin** | basicDataTypes:Boolean | 0..1 | The fin property If this flag is set, it means there is no more data from sender. |

## 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 3‑38**.

Table ‑. Properties of the UDPHeaderType class

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

## IANAHardwareType Class

IANAHardwareType class specifies the class of hardware, via a union of the IANAHardwareTypeEnum enumeration and the atomic xs:string type. Its base type is the BaseObjectPropertyType class, for permitting complex (i.e. regular-expression based) specifications.

## IANAEtherType Class

EtherObjectType class specifies "class" field of Ethernets, via a union of the IANAEtherTypeEnum enumeration and the atomic xs:string type. Its base type is the BaseObjectPropertyType class, for permitting complex (i.e. regular-expression based) specifications.

## IANAAssignedIPNumbersType Class

IANAAssignedIPNumbersType class specifies Internet Protocol numbers, via a union of the IANAAssignedIPNumbersTypeEnum enumeration and the atomic xs:string type. Its base type is the BaseObjectPropertyType class, for permitting complex (i.e. regular-expression based) specifications.

## IANAPortNumberRegistryType Class

IANAPortNumberRegistryType class specifies port numbers, via a union of the IANAPortNumberRegistryTypeEnum enumeration and the atomic xs:string type. Its base type is the BaseObjectPropertyType class, for permitting complex (i.e. regular-expression based) specifications.

## ICMPv4PacketType Class

ICMP is used to send error messages (e.g., a datagram cannot reach its destination), informational messages ( e.g., timestamp information), or a traceroute message. REF: http://www.networksorcery.com/enp/protocol/icmp.htm.

The property table of the ICMPv4PacketType class is given in **Table 3‑39**.

Table ‑. Properties of the ICMPv4PacketType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **ICMPv4\_Header** | PacketObj:  ICMPv4HeaderType | 0..1 | The ICMPv4\_Header property Actual header bytes are captured here. The message content of each type/code pair is also defined as part of the larger, complex "ICMPv4PacketType" type as either an error message, an informational message, or a traceroute message. The meaning of the type and code bytes is made explicit in the elements corresponding to each message type. |
| **Error\_Msg** | PacketObj:  ICMPv4ErrorMessageType | 0..1 | For ICMP error messages, boolean values are used in this property to explicitly interpret the type and code bytes appearing in the ICMP header. Additional fields and message content are also defined here. |
| **Info\_Msg** | PacketObj:  ICMPv4InfoMessageType | 0..1 | For ICMP informational messages, boolean values are used in this property to explicitly interpret the type and code bytes appearing in the ICMP header. Additional fields and message content are also defined here. |
| **Traceroute** | PacketObj:  ICMPv4TracerouteType | 0..1 | For ICMP traceroute messages (type = 30), specifies related propertys and ICMP code value. A boolean value is used to explicitly interpret the code byte appearing in the ICMP header. Additional fields and message content are also defined here. |

## ICMPv4HeaderType Class

Actual ICMP header bytes are defined, corresponding to the ICMP class, ICMP code, and to the checksum.

The property table of the ICMPv4HeaderType class is given in **Table 3‑40**.

Table ‑. Properties of the ICMPv4HeaderType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Type** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Type property ICMP Type byte specifies the format of the ICMP message. |
| **Code** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Code property ICMP Code byte further qualifies the ICMP message. |
| **Checksum** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Checksum property ICMP Checksum (16 bits) covers the ICMP message. |

## ICMPv4ErrorMessageType Class

ICMP error messages include destination unreachable messages, source quench messages, redirect messages, and time exceeded messages.

The property table of the ICMPv4ErrorMessageType class is given in **Table 3‑41**.

Table ‑. Properties of the ICMPv4ErrorMessageType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Destination\_Unreachable** | PacketObj:  ICMPv4DestinationUnreachableType | 0..1 | The Destination\_Unreachable property A destination unreachable message is an ICMP message which is generated by the host or its inbound gateway to inform the client that the destination is unreachable for some reason (http://en.wikipedia.org/wiki/ICMP\_Destination\_Unreachable). |
| **Source\_Quench** | PacketObj:  ICMPv4SourceQuenchType | 0..1 | The Source\_Quench property A source quench message is an ICMP message that requests that 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 its limit (http://en.wikipedia.org/wiki/ICMP\_Source\_Quench). |
| **Redirect\_Message** | PacketObj:  ICMPv4RedirectMessageType | 0..1 | The Redirect\_Message property A redirect message is used to send data packets on an alternative route. This ICMP redirect message informs a host to update its routing information. |
| **Time\_Exceeded** | PacketObj:  ICMPv4TimeExceededType | 0..1 | An ICMP time exceeded message 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 (http://en.wikipedia.org/wiki/ICMP\_Time\_Exceeded). |
| **Error\_Msg\_Content** | PacketObj:  ICMPv4ErrorMessageContentType | 0..1 | The Error\_Msg\_Content property Message content common to all ICMP error messages are defined here. Fields that are specific to individual messages are defined separately under each message type. |

## ICMPv4ErrorMessageContentType Class

Elements associated with ICMPv4 error messages (as opposed to ICMP informational messages or ICMP traceroute message).

The property table of the ICMPv4ErrorMessageContentType class is given in **Table 3‑42**.

Table ‑. Properties of the ICMPv4ErrorMessageContentType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **IP\_Header** | PacketObj:IPv4HeaderType | 0..1 | The IP\_Header property IP header from the original datagram. |
| **First\_Eight\_Bytes** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The First\_Eight\_Bytes property First 8 bytes of the original datagram's data. |

## ICMPv4InfoMessageType Class

ICMP informational messages include echo request/reply, timestamp request/reply, and address mask request/reply.

The property table of the ICMPv4InfoMessageType class is given in **Table 3‑43**.

Table ‑. Properties of the ICMPv4InfoMessageType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Echo\_Reply** | PacketObj:  ICMPv4EchoReplyType | 0..1 | Echo reply/request messages are also known as "ping". The Info\_Message\_Content property contains an identifier and sequence number which together form the "quench" for echo reply and echo request. Fields specific to an echo reply message are given as elements to this echo reply field (type=0). |
| **Echo\_Request** | PacketObj:  ICMPv4EchoRequestType | 0..1 | Echo reply/request messages are also known as "ping". The Info\_Message\_Content property contains an identifier and sequence number which together form the "quench" for echo reply and echo request. Fields specific to an echo request message are given as elements to this echo request field (type=8). |
| **Timestamp\_Request** | PacketObj:  ICMPv4TimestampRequestType | 0..1 | The Timestamp\_Request property A timestamp request is an ICMP informational message used for time synchronization. |
| **Timestamp\_Reply** | PacketObj:  ICMPv4TimestampReplyType | 0..1 | The Timestamp\_Reply property A timestamp reply is an informational ICMP message which replies to a timestamp request message. |
| **Address\_Mask\_Request** | PacketObj:  ICMPv4AddressMaskRequestType | 0..1 | The Address\_Mask\_Request property An address mask request is an ICMP informational message (query message) normally sent by a host to a router in order to obtain an appropriate subnet mask (type=17). |
| **Address\_Mask\_Reply** | PacketObj:  ICMPv4AddressMaskReplyType | 0..1 | The Address\_Mask\_Reply property An address mask reply is an ICMP informational message, used to reply to an address mask request message with an appropriate subnet mask (type=18). |
| **Info\_Msg\_Content** | PacketObj:  ICMPv4InfoMessageContentType | 0..1 | The Info\_Msg\_Content property Fields that are common to all ICMP informational messages are defined here. Fields that are specific to individual messages are defined separately under each message type. |

## ICMPv4InfoMessageContentType Class

Elements associated with ICMPv4 informational messages (as opposed to ICMP error messages or ICMP traceroute message).

The property table of the ICMPv4InfoMessageContentType class is given in **Table 3‑44**.

Table ‑. Properties of the ICMPv4InfoMessageContentType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Identifier** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Identifier property 16-bit identifier. Combined with the sequence number, called the "quench" for echo reply and echo request. |
| **Sequence\_Number** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Sequence\_Number property 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. |

## ICMPv4TracerouteType Class

Elements associated with ICMPv4 traceroute message (as opposed to ICMP error messages or ICMP informational messages); corresponds to ICMP class =30. (http://www.networksorcery.com/enp/protocol/icmp/msg30.htm).

The property table of the ICMPv4TracerouteType class is given in **Table 3‑45**.

Table ‑. Properties of the ICMPv4TracerouteType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Outbound\_Packet\_**  **Forward\_Success** | basicDataTypes:Boolean | 0..1 | The Outbound\_Packet\_Forward\_Success property One of two possible subtypes for an ICMP traceroute message. This subtype means that the outbound packet was successfully forwarded (code=0). |
| **Outbound\_Packet\_**  **no\_Route** | basicDataTypes:Boolean | 0..1 | The Outbound\_Packet\_no\_Route property One of two possible subtypes for an ICMP traceroute message. This one means that there is no route for the outbound packet and the packet was discarded (code=1). |
| **Identifier** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Identifier property 16 bits. 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). (http://www.networksorcery.com/enp/protocol/icmp/msg30.htm). |
| **Outbound\_Hop\_Count** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Outbound\_Hop\_Count property 16 bits. Outbound hop count as copied from the IP traceroute option of the packet which caused this traceroute message to be sent (http://www.networksorcery.com/enp/protocol/icmp/msg30.htm). |
| **Return\_Hop\_Count** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Return\_Hop\_Count property 16 bits. Return hop count as copied from the IP traceroute options of the packet which caused this traceroute message to be sent. (http://www.networksorcery.com/enp/protocol/icmp/msg30.htm). |
| **Output\_Link\_Speed** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | 32 bits. 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. (http://www.networksorcery.com/enp/protocol/icmp/msg30.htm). |
| **Output\_Link\_MTU** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | 32 bits. 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. (http://www.networksorcery.com/enp/protocol/icmp/msg30.htm). |

## ICMPv6PacketType Class

ICMP is used to send error messages (e.g., a datagram cannot reach its destination), informational messages ( e.g., ping). Only the message classs defined in RFC 4443 (ICMP v6) are included; additional message types will be defined as needed. REF: http://tools.ietf.org/html/rfc4443 and http://www.networksorcery.com/enp/protocol/icmpv6.htm and http://en.wikipedia.org/wiki/ICMPv6.

The property table of the ICMPv6PacketType class is given in **Table 3‑46**.

Table ‑. Properties of the ICMPv6PacketType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **ICMPv6\_Header** | PacketObj:  ICMPv6HeaderType | 0..1 | The ICMPv6\_Header property Actual ICMP v6 header bytes are defined, corresponding to the ICMP type, ICMP code, and to the checksum. |
| **Error\_Msg** | PacketObj:  ICMPv6ErrorMessageType | 0..1 | For ICMP v6 error messages, boolean values are used in this property to explicitly interpret the type and code bytes appearing in the ICMP header. Additional fields and message content are also defined here. The type value indicates whether an ICMP message is an error message (type is 0 to 127) or an information message (type is 128 to 255). |
| **Info\_Msg** | PacketObj:  ICMPv6InfoMessageType | 0..1 | For ICMP v6 informational messages, boolean values are used in this property to explicitly interpret the type and code bytes appearing in the ICMP header. Additional fields and message content are also defined here. The type value indicates whether an ICMP message is an error message (type is 0 to 127) or an information message (type is 128 to 255). |

## ICMPv6HeaderType Class

Actual ICMP header bytes are defined, corresponding to the ICMP class, ICMP code, and to the checksum. Translation of each type and code byte are defined in text by using boolean values associated with corresponding elements in the informational and error message type elements.

The property table of the ICMPv6HeaderType class is given in **Table 3‑47**.

Table ‑. Properties of the ICMPv6HeaderType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Type** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Type property The ICMP v6 type byte 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** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Code property The code byte value depends on the message type and provides an additional level of message granularity. |
| **Checksum** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Checksum property Checksum characterizes the checksum information of an ICMPv6 header. |

## ICMPv6ErrorMessageType Class

ICMP v6 error messages include destination unreachable messages, packet too big messages, and time exceeded messages, and parameter problem messages, as defined in RFC 2463. Type values of ICMP v6 error messages range from 1 to 127.

The property table of the ICMPv6ErrorMessageType class is given in **Table 3‑48**.

Table ‑. Properties of the ICMPv6ErrorMessageType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Destination\_Unreachable** | PacketObj:  ICMPv6DestinationUnreachableType | 0..1 | The Destination\_Unreachable property A destination unreachable message 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. (http://tools.ietf.org/html/rfc4443). |
| **Packet\_Too\_Big** | PacketObj:  ICMPv6PacketTooBigType | 0..1 | The Packet\_Too\_Big property A packet too big message 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. |
| **Time\_Exceeded** | PacketObj:  ICMPv6TimeExceededType | 0..1 | The Time\_Exceeded property A time exceeded message is send if either the hop limit is exceeded (hop limit = 0) or if fragment reassembly has timed out. |
| **Parameter\_Problem** | PacketObj:  ICMPv6ParameterProblemType | 0..1 | 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 of the packet, it should send an ICMPv6 Parameter Problem message to the packet's source (http://tools.ietf.org/html/rfc4443). |
| **Invoking\_Packet** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Invoking\_Packet property as much of invoking packet as possible without the ICMPv6 packet exceeding the minimum IPc6 MTU. |

## ICMPv6InfoMessageType Class

ICMP v6 informational messages include echo request/reply; other informational message classs will be added in the future as they are more commonly used (only echo request/reply are defined in RFC 4443).

The property table of the ICMPv6InfoMessageType class is given in **Table 3‑49**.

Table ‑. Properties of the ICMPv6InfoMessageType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Echo\_Request** | PacketObj:  ICMPv6EchoRequestType | 0..1 | The Echo\_Request property Echo request and reply messages are used for diagnostic purposes. |
| **Echo\_Reply** | PacketObj:  ICMPv6EchoReplyType | 0..1 | The Echo\_Reply property Echo request and reply messages are used for diagnostic purposes. |
| **Info\_Msg\_Content** | PacketObj:  ICMPv6InfoMessageContentType | 0..1 | The Info\_Msg\_Content property Fields that are common to all ICMP v6 informational messages are defined here. Fields that are specific to individual messages are defined separately under each message type. |

## ICMPv6InfoMessageContentType Class

Elements associated with ICMPv6 informational messages (as opposed to ICMP v6 error messages).

The property table of the ICMPv6InfoMessageContentType class is given in **Table 3‑50**.

Table ‑. Properties of the ICMPv6InfoMessageContentType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Identifier** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Identifier property 16-bit identifier. Combined with the sequence number, called the "quench" for echo reply and echo request. |
| **Sequence\_Number** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Sequence\_Number property 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. |

## ICMPv4EchoReplyType Class

Echo reply v4 informational message (used to ping); ICMP class=0.

The property table of the ICMPv4EchoReplyType class is given in **Table 3‑51**.

Table ‑. Properties of the ICMPv4EchoReplyType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Echo\_Reply** | basicDataTypes:Boolean | 0..1 | The Echo\_Reply property Echo reply is the only subtype (code=0). |
| **Data** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Data property This data is optional and is used for the different kind of answers given with an ICMP Echo Reply message. Can be arbitrary length (but less than the MTU of the network). |

## ICMPv4DestinationUnreachableType Class

Destination Unreachable error message; ICMP class=3.

The property table of the ICMPv4DestinationUnreachableType class is given in **Table 3‑52**.

Table ‑. Properties of the ICMPv4DestinationUnreachableType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Destination\_Network\_Unreachable** | basicDataTypes:Boolean | 0..1 | The Destination\_Network\_Unreachable property One of 16 different subtypes of a destination unreachable ICMP message; destination network unreachable (code=0). |
| **Destination\_Host\_Unreachable** | basicDataTypes:Boolean | 0..1 | The Destination\_Host\_Unreachable property One of 16 different subtypes of a destination unreachable ICMP message; destination host unreachable (code=1). |
| **Destination\_Protocol\_Unreachable** | basicDataTypes:Boolean | 0..1 | The Destination\_Protocol\_Unreachable property One of 16 different subtypes of a destination unreachable ICMP message; destination protocol unreachable (code=2). |
| **Destination\_Port\_Unreachable** | basicDataTypes:Boolean | 0..1 | The Destination\_Port\_Unreachable property One of 16 different subtypes of a destination unreachable ICMP message; destination port unreachable (code=3). |
| **Fragmentation\_Required** | PacketObj:  FragmentationRequiredType | 0..1 | One of 16 different subtypes of a destination unreachable ICMP message; fragmentation required (code=4). This property has an additional field (Next-Hop MTU), as well as a boolean value indicating this subtype. |
| **Source\_Route\_Failed** | basicDataTypes:Boolean | 0..1 | The Source\_Route\_Failed property One of 16 different subtypes of a destination unreachable ICMP message; source route failed (code=5). |
| **Destination\_Network\_Unknown** | basicDataTypes:Boolean | 0..1 | The Destination\_Network\_Unknown property One of 16 different subtypes of a destination unreachable ICMP message; destination network unknown (code=6). |
| **Destination\_Host\_Unknown** | basicDataTypes:Boolean | 0..1 | The Destination\_Host\_Unknown property One of 16 different subtypes of a destination unreachable ICMP message; destination host unknown (code=7). |
| **Source\_Host\_Isolated** | basicDataTypes:Boolean | 0..1 | The Source\_Host\_Isolated property One of 16 different subtypes of a destination unreachable ICMP message; source host isolated (code=8). |
| **Network\_Administratively\_**  **Prohibited** | basicDataTypes:Boolean | 0..1 | The Network\_Administratively\_Prohibited property One of 16 different subtypes of a destination unreachable ICMP message; host administratively prohibited (code=9). |
| **Host\_Administratively\_Prohibited** | basicDataTypes:Boolean | 0..1 | The Host\_Administratively\_Prohibited property One of 16 different subtypes of a destination unreachable ICMP message; host administratively prohibited (code=10). |
| **Network\_Unreachable\_For\_TOS** | basicDataTypes:Boolean | 0..1 | The Network\_Unreachable\_For\_TOS property One of 16 different subtypes of a destination unreachable ICMP message; network unreachable for TOS (code=11). |
| **Host\_Unreachable\_For\_TOS** | basicDataTypes:Boolean | 0..1 | The Host\_Unreachable\_For\_TOS property One of 16 different subtypes of a destination unreachable ICMP message; host unreachable for TOS (code=12). |
| **Communication\_Administratively\_**  **Prohibited** | basicDataTypes:Boolean | 0..1 | The Communication\_Administratively\_Prohibited property One of 16 different subtypes of a destination unreachable ICMP message; communication administratively prohibited (code=13). |
| **Host\_Precedence\_Violation** | basicDataTypes:Boolean | 0..1 | The Host\_Precedence\_Violation property One of 16 different subtypes of a destination unreachable ICMP message; host precedence violation (code=14). |
| **Precedence\_Cutoff\_In\_Effect** | basicDataTypes:Boolean | 0..1 | The Precedence\_Cutoff\_In\_Effect property One of 16 different subtypes of a destination unreachable ICMP message; precedence cutoff in effect (code=15). |

## FragmentationRequiredType Class

This further specifies an ICMP destination unreachable (class=3) message of code=4 (fragmentation required) message by providing a Next-Hop MTU field.

The property table of the FragmentationRequiredType class is given in **Table 3‑53**.

Table ‑. Properties of the FragmentationRequiredType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Fragmentation\_Required** | basicDataTypes:Boolean | 0..1 | The Fragmentation\_Required property Indicates that the subtype of the destination unreachable ICMP message is "fragmentation required". |
| **Next\_Hop\_MTU** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Next-Hop MTU property contains the MTU of the next-hop network is a code 4 error (fragmentation required) occurs. |

## ICMPv4SourceQuenchType Class

Source Quench (congestion control) error message; ICMP class=4.

The property table of the ICMPv4SourceQuenchType class is given in **Table 3‑54**.

Table ‑. Properties of the ICMPv4SourceQuenchType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Source\_Quench** | basicDataTypes:Boolean | 0..1 | The Source\_Quench property Source quench is the only subtype (code=0). |

## ICMPv4RedirectMessageType Class

Redirect Message error message; ICMP class=5.

The property table of the ICMPv4RedirectMessageType class is given in **Table 3‑55**.

Table ‑. Properties of the ICMPv4RedirectMessageType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Network\_Redirect** | basicDataTypes:  Boolean | 0..1 | The Network\_Redirect property One of 4 different subtypes of a redirect ICMP message; redirect datagram for the network (code=0). |
| **Host\_Redirect** | basicDataTypes:  Boolean | 0..1 | The Host\_Redirect property One of 4 different subtypes of a redirect ICMP message; redirect datagram for the host (code=1). |
| **ToS\_Network\_Redirect** | basicDataTypes:  Boolean | 0..1 | The ToS\_Network\_Redirect property One of 4 different subtypes of a redirect ICMP message; redirect datagram for the TOS and network (code=2). |
| **ToS\_Host\_Redirect** | basicDataTypes:  Boolean | 0..1 | The ToS\_Host\_Redirect property One of 4 different subtypes of a redirect ICMP message; redirect datagram for the TOS and host (code=3). |
| **IP\_Address** | AddressObj:  AddressObjectType | 0..1 | The IP\_Address property The IP address is the 32-bit address of the gateway to which the redirection should be sent. |

## ICMPv4EchoRequestType Class

Echo Request informational message (used to ping); ICMP class=8.

The property table of the ICMPv4EchoRequestType class is given in **Table 3‑56**.

Table ‑. Properties of the ICMPv4EchoRequestType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Echo\_Request** | basicDataTypes:Boolean | 0..1 | The Echo\_Request property Echo request is the only subtype (code=0). |
| **Data** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Data property This data is optional and is used for the different kind of answers given with an ICMP Echo Request message. Can be arbitrary length (but less than the MTU of the network). |

## ICMPv4TimeExceededType Class

Time Exceeded error message; ICMP class=11.

The property table of the ICMPv4TimeExceededType class is given in **Table 3‑57**.

Table ‑. Properties of the ICMPv4TimeExceededType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **TTL\_Exceeded\_In\_Transit** | basicDataTypes:  Boolean | 0..1 | The TTL\_Exceeded\_In\_Transit property specifies that the time-to-live was exceeded in transit (code=0). |
| **Frag\_Reassembly\_**  **Time\_Exceeded** | basicDataTypes:  Boolean | 0..1 | The Frag\_Reassembly\_Time\_Exceeded property specifies that the fragment reassembly time was exceeded (code=1). |

## ICMPv4TimestampRequestType Class

Time Stamp Request informational message; ICMP class=13.

The property table of the ICMPv4TimestampRequestType class is given in **Table 3‑58**.

Table ‑. Properties of the ICMPv4TimestampRequestType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Timestamp** | basicDataTypes:Boolean | 0..1 | The Timestamp property This is the only subtype of a timestamp request message (code=0). |
| **Originate\_Timestamp** | cyboxCommon:  UnsignedIntegerObjectPropertyType | 0..1 | The Originate\_Timestamp property 32-bits; number of ms since midnight UT. 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. |

## ICMPv4TimestampReplyType Class

Time Stamp Reply informational message; ICMP class=14.

The property table of the ICMPv4TimestampReplyType class is given in **Table 3‑59**.

Table ‑. Properties of the ICMPv4TimestampReplyType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Timestamp\_Reply** | basicDataTypes:Boolean | 0..1 | The Timestamp\_Reply property This is the only subtype of a timestamp reply message (code=0). |
| **Originate\_Timestamp** | cyboxCommon:  UnsignedIntegerObjectPropertyType | 0..1 | The Originate\_Timestamp property 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. |
| **Receive\_Timestamp** | cyboxCommon:  UnsignedIntegerObjectPropertyType | 0..1 | The Receive\_Timestamp property The receive timestamp is 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 Transmit\_Timestamp property The transmit timestamp is 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. |

## ICMPv4AddressMaskRequestType Class

Address Mask Request informational message; ICMP class=17.

The property table of the ICMPv4AddressMaskRequestType class is given in **Table 3‑60**.

Table ‑. Properties of the ICMPv4AddressMaskRequestType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Address\_Mask\_Request** | basicDataTypes:Boolean | 0..1 | The Address\_Mask\_Request property This is the only possible subtype of an address mask request message (code=0). |
| **Address\_Mask** | AddressObj:  AddressObjectType | 0..1 | The Address\_Mask property The address mask can be set to 0 in an address mask request message (as opposed to an address mask reply message, in which case it should be set to the subnet mask). |

## ICMPv4AddressMaskReplyType Class

Address Mask informational message; ICMP class=18.

The property table of the ICMPv4AddressMaskReplyType class is given in **Table 3‑61**.

Table ‑. Properties of the ICMPv4AddressMaskReplyType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Address\_Mask\_Reply** | basicDataTypes:Boolean | 0..1 | The Address\_Mask\_Reply property This is the only possible subtype of an address mask reply message (code=0). |
| **Address\_Mask** | AddressObj:  AddressObjectType | 0..1 | This address mask property should be set to the subnet mask. |

## ICMPv6DestinationUnreachableType Class

Destination unreachable error message; ICMP v6 class=1.

The property table of the ICMPv6DestinationUnreachableType class is given in **Table 3‑62**.

Table ‑. Properties of the ICMPv6DestinationUnreachableType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **No\_Route** | basicDataTypes:Boolean | 0..1 | The No\_Route property No route to destination (ICMP v6 code=0). |
| **Comm\_Prohibited** | basicDataTypes:Boolean | 0..1 | The Comm\_Prohibited property Communication with destination administratively prohibited (ICMP v6 code=1). |
| **Beyond\_Scope** | basicDataTypes:Boolean | 0..1 | The Beyond\_Scope property Beyond scope of source address (ICMP v6 code =2). |
| **Address\_Unreachable** | basicDataTypes:Boolean | 0..1 | The Address\_Unreachable property Address is unreachable (ICMP v6 code=3). |
| **Port\_Unreachable** | basicDataTypes:Boolean | 0..1 | The Port\_Unreachable property Port is unreachable (ICMP v6 code=4). |
| **Src\_Addr\_Failed\_Policy** | basicDataTypes:Boolean | 0..1 | The Src\_Addr\_Failed\_Policy property Source address failed ingress/egress policy (ICMP v6 code=5). |
| **Reject\_Route** | basicDataTypes:Boolean | 0..1 | The Reject\_Route property Reject route to destination (ICMP v6 code=6). |

## ICMPv6PacketTooBigType Class

Packet too big error message; ICMP v6 class=2.

The property table of the ICMPv6PacketTooBigType class is given in **Table 3‑63**.

Table ‑. Properties of the ICMPv6PacketTooBigType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Packet\_Too\_Big** | basicDataTypes:Boolean | 0..1 | The Packet\_Too\_Big property Only one code value is defined and is set to 0 (zero) by the originator and ignored by the receiver. |
| **MTU** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The MTU property Maximum Transmission Unit describes the size limit for any given physical network. |

## ICMPv6TimeExceededType Class

Time exceeded error message; ICMP v6 class=3.

The property table of the ICMPv6TimeExceededType class is given in **Table 3‑64**.

Table ‑. Properties of the ICMPv6TimeExceededType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Hop\_Limit\_Exceeded** | basicDataTypes:Boolean | 0..1 | The Hop\_Limit\_Exceeded property Hop limit exceeded in transit (ICMP v6 code=0). |
| **Fragment\_Reassem\_Time\_Exceeded** | basicDataTypes:Boolean | 0..1 | The Fragment\_Reassem\_Time\_Exceeded property Fragment reassembly time exceeded (ICMP v6 code=1). |

## ICMPv6ParameterProblemType Class

Parameter problem error message; ICMP v6 class=4.

The property table of the ICMPv6ParameterProblemType class is given in **Table 3‑65**.

Table ‑. Properties of the ICMPv6ParameterProblemType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Erroneous\_Header\_Field** | basicDataTypes:Boolean | 0..1 | Erroneous header property encountered (ICMP v6 code=0). |
| **Unrecognized\_Next\_**  **Header\_Type** | basicDataTypes:Boolean | 0..1 | The Unrecognized\_Next\_Header\_Type property Unrecognized next header type encountered (ICMP v6 code=1). |
| **Unrecognized\_IPv6\_Option** | basicDataTypes:Boolean | 0..1 | The Unrecognized\_IPv6\_Option property Unrecognized IP v6 option encountered (ICMP v6 code=2). |
| **Pointer** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Pointer property identifies octet offset within invoking packet where error was detected. |

## ICMPv6EchoRequestType Class

Echo request informational ICMP v6 message; class=128.

The property table of the ICMPv6EchoRequestType class is given in **Table 3‑66**.

Table ‑. Properties of the ICMPv6EchoRequestType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Echo\_Request** | basicDataTypes:Boolean | 0..1 | The Echo\_Request property Every node must implement an ICMP v6 Echo responder function that receives Echo Requests (ICMP v6 code=0). |
| **Data** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Data property Zero or more octets of arbitrary data. |

## ICMPv6EchoReplyType Class

Echo reply informational ICMP v6 message; class=129.

The property table of the ICMPv6EchoReplyType class is given in **Table 3‑67**.

Table ‑. Properties of the ICMPv6EchoReplyType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Echo\_Reply** | basicDataTypes:Boolean | 0..1 | The Echo\_Reply property Every node must implement an ICMP v6 Echo responder function that originates corresponding Echo Replies(ICMP v6 code=0). |
| **Data** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Data property This is the data from the invoking echo request message. |

## PrefixType Class

Provides an IP address or a prefix of an IP address for NDP for IPv6.

The property table of the PrefixType class is given in **Table 3‑68**.

Table ‑. Properties of the PrefixType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **IPv6\_Addr** | AddressObj:AddressObjectType | 0..1 | The IPv6\_Addr property IPv6 address. |
| **IP\_Addr\_Prefix** | AddressObj:AddressObjectType | 0..1 | The IP\_Addr\_Prefix property The initial bits of an IPv6 address (these are identical for all hosts in a network) form the network's prefix. http://ipv6.com/articles/general/IPv6-Addressing.htm. |

## HopByHopOptionsType Class

Defines fields for 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 3‑69**.

Table ‑. Properties of the HopByHopOptionsType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Next\_Header** | PacketObj:  IANAAssignedIPNumbersType | 0..1 | Identifies the type of header immediately following the Hop-by-Hop Options header. Uses the same values as the IPv4 Protocol property. |
| **Header\_Ext\_Len** | cyboxCommon: HexBinaryObjectPropertyType | 0..1 | The Header\_Ext\_Len property Length of the Hop-by-Hop Options header in 8-octet units, not including the first 8 octets. |
| **Option\_Data** | PacketObj:  OptionDataType | 0..\* | 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. |

## OptionDataType Class

Defines 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 3‑70**.

Table ‑. Properties of the OptionDataType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Option\_Type** | PacketObj:IPv6OptionType | 0..1 | The Option\_Type property Identifies the type of option. This 8-bit Option Type identifier is internally encoded such that different bits have different meanings. These meanings are further specified in the IPv6OptionType type. |
| **Option\_Data\_Len** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | Length of the Option Data property of this option, in octets. |
| **Pad1** | PacketObj:Pad1Type | 0..1 | The Pad1 option is used to insert one octet of padding into the Options area of a header. The Pad1 option does not have length and value propertys. |
| **PadN** | PacketObj:PadNType | 0..1 | The PadN property The PadN option is used to insert two or more octets of paddings into the Options area of a header. |

## RoutingType Class

Specifies the fields 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. http://tools.ietf.org/html/rfc2460.

The property table of the RoutingType class is given in **Table 3‑71**.

Table ‑. Properties of the RoutingType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Next\_Header** | PacketObj:  IANAAssignedIPNumbersType | 0..1 | Identifies 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 Header\_Ext\_Len property length of the Routing header in 8-octet units, not including the first 8 octets. |
| **Routing\_Type** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Routing\_Type property 8-bit identifiers of a particular Routing header variant. Further definition will be added as required. |
| **Segments\_Left** | cyboxCommon:  IntegerObjectPropertyType | 0..1 | The Segments\_Left property Number of route segments remaining, i.e., number of explicitly listed intermediate nodes still to be visited before reaching the final destination. http://tools.ietf.org/html/rfc2460. |
| **Type\_Specific\_Data** | cyboxCommon:  StringObjectPropertyType | 0..1 | Variable length property, of format determined by the Routing Type. |

## FragmentType Class

Specifies the fields of the Fragment header, which is used by an IPv6 source to send a packet larger than would fit in the path MTU. http://tools.ietf.org/html/rfc2460.

The property table of the FragmentType class is given in **Table 3‑72**.

Table ‑. Properties of the FragmentType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Fragment\_Header** | PacketObj:  FragmentHeaderType | 0..1 | The Fragment\_Header property Each fragment has 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 Fragment property 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. |

## DestinationOptionsType Class

Defines fields 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 3‑73**.

Table ‑. Properties of the DestinationOptionsType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Next\_Header** | PacketObj:  IANAAssignedIPNumbersType | 0..1 | Identifies the type of header immediately following the Destination\_Options options header. Uses the same values as the IPv4 Protocol property. |
| **Header\_Ext\_Len** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Header\_Ext\_Len property Length of the Destination Options header in 8-octet units, not including the first 8 octets. |
| **Option\_Data** | PacketObj:OptionDataType | 0..\* | Variable-length property, of length such that the complete Destinations Options header is an integer multiple of 8 octets long. Contains one or more type-length-value (TLV)-encoded options. |

## AuthenticationHeaderType Class

The IP Authentication Header is used to provide connectionless integrity and data origin authentication for IP datagrams and to provide protection against replays. http://www.ietf.org/rfc/rfc2402.txt.

The property table of the AuthenticationHeaderType class is given in **Table 3‑74**.

Table ‑. Properties of the AuthenticationHeaderType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Next\_Header** | PacketObj:  IANAAssignedIPNumbersType | 0..1 | Identifies 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 | An 8-bit property specifying the length of the AH in 32-bit words. |
| **Security\_Parameters\_Index** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Security\_Parameters\_Index property The SPI 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. http://www.ietf.org/rfc/rfc2402.txt. |
| **Sequence\_Number** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | This unsigned 32-bit property contains a monotonically increasing counter value (sequence number). |
| **Authentication\_Data** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | This is 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. |

## EncapsulatingSecurityPayloadType Class

ESP 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. http://www.ietf.org/rfc/rfc2406.txt.

The property table of the EncapsulatingSecurityPayloadType class is given in **Table 3‑75**.

Table ‑. Properties of the EncapsulatingSecurityPayloadType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Security\_Parameters\_**  **Index** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Security\_Parameters\_Index property The SPI 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. http://www.ietf.org/rfc/rfc2406.txt. |
| **Sequence\_Number** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | This unsigned 32-bit property contains a monotonically increasing counter value (sequence number). |
| **Payload\_Data** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | Payload Data is a variable-length property containing data described by the Next Header field. |
| **Padding** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The padding property 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 Padding\_Len property The pad length indicates the number of pad bytes immediately preceding it. Range is 0-255, where a value of zero indicates that no padding bytes are present. http://www.ietf.org/rfc/rfc2406.txt. |
| **Next\_Header** | PacketObj:  IANAAssignedIPNumbersType | 0..1 | Identifies the type data contained in the payload data property. Uses the same values as the IPv4 Protocol field. |
| **Authentication\_Data** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Authentication Data is a variable-length property containing an Integrity Check Value (ICV) computed over the ESP packet minus the Authentication Data. http://www.ietf.org/rfc/rfc2406.txt. |

## Pad1Type Class

The Pad1 class specifies how 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 3‑76**.

Table ‑. Properties of the Pad1Type class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Octet** | cyboxCommon:  HexBinaryObjectPropertyType | 1 | The Octet property The fixed 00 value specifies that the Pad1 option is used and also serves as the single octet of padding. |

## PadNType Class

The PadN class specifies how 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 3‑77**.

Table ‑. Properties of the PadNType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Octet** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Octet property Specifies the PandN option. |
| **Option\_Data\_Length** | cyboxCommon:  IntegerObjectPropertyType | 0..1 | Length of the padding. For N octets of padding, the Option\_Data\_Length propertys contains the value N-2. |
| **Option\_Data** | cyboxCommon:  IntegerObjectPropertyType | 0..1 | The Option\_Data property Actual padding; consists of N-2 zero-valued octets. |

## FragmentHeaderType Class

Each fragment has 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 3‑78**.

Table ‑. Properties of the FragmentHeaderType class

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Multiplicity** | **Description** |
| **Next\_Header** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | Identifies the type of header immediately following the Fragment header. Uses the same values as the IPv4 Protocol property. |
| **Fragment\_Offset** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Fragment\_Offset property 13-bit unsigned integer. 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** | PacketObj:MFlagType | 0..1 | The M\_Flag property Indicates whether this is the last fragment or whether there are more fragments. |
| **Identification** | cyboxCommon:  HexBinaryObjectPropertyType | 0..1 | The Identification property For every packet that is to be fragmented, the source node generates a 32-bit Identification value. |

## MFlagType Class

MFlagType class specifies whether there are more fragments, via a union of the MFlagTypeEnum enumeration and the atomic xs:string type. Its base type is the BaseObjectPropertyType class, for permitting complex (i.e. regular-expression based) specifications.

## ARPOpTypeEnum Enumeration

The literals of the ARPOpTypeEnum enumeration are given in **Table 3‑79**.

Table ‑. 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. |

## DoNotFragmentTypeEnum Enumeration

The literals of the DoNotFragmentTypeEnum enumeration are given in **Table 3‑80**.

Table ‑. Literals of the DoNotFragmentTypeEnum enumeration

|  |  |
| --- | --- |
| **Enumeration Literal** | **Description** |
| **fragementifnecessary(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. |

## MoreFragmentsTypeEnum Enumeration

The literals of the MoreFragmentsTypeEnum enumeration are given in **Table 3‑81**.

Table ‑. Literals of the MoreFragmentsTypeEnum enumeration

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

## IPv4CopyFlagTypeEnum Enumeration

The literals of the IPv4CopyFlagTypeEnum enumeration are given in **Table 3‑82**.

Table ‑. Literals of the IPv4CopyFlagTypeEnum enumeration

|  |  |
| --- | --- |
| **Enumeration Literal** | **Description** |
| **donotcopy(0)** | Indicates that the options need NOT be copied into all fragments of a fragmented packet. |
| **copy(1)** | Indicates that the options need to be copied into all fragments of a fragmented packet. |

## IPv4ClassTypeEnum Enumeration

The literals of the IPv4ClassTypeEnum enumeration are given in **Table 3‑83**.

Table ‑. 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. |

## IPv4OptionsTypeEnum Enumeration

The literals of the IPv4OptionsTypeEnum enumeration are given in **Table 3‑84**.

Table ‑. 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)** |  |
| **imitrafficdescriptor(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. |
| **selectivedirectedbroadcasemode(21)** | Indicates a Selective Directed Broadcast option, or SDB. |
| **dynamicepacketstate(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. |

## IPv6DoNotRecogActionTypeEnum Enumeration

The literals of the IPv6DoNotRecogActionTypeEnum enumeration are given in **Table 3‑85**.

Table ‑. Literals of the IPv6DoNotRecogActionTypeEnum enumeration

|  |  |
| --- | --- |
| **Enumeration Literal** | **Description** |
| **skipoption(00)** | Indicates that the option should be skipped and the header should continue to be processed. See RFC 2460. |
| **discardpacket(01)** | Indicates that the packet should be discarded. See RFC 2460. |
| **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. See RFC 2460. |
| **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 Type. See RFC 2460. |

## IPv6PacketChangeTypeEnum Enumeration

The literals of the IPv6PacketChangeTypeEnum enumeration are given in **Table 3‑86**.

Table ‑. Literals of the IPv6PacketChangeTypeEnum enumeration

|  |  |
| --- | --- |
| **Enumeration Literal** | **Description** |
| **nochange(0)** | Indicates that the packet does not change en-route. See RFC 2460. |
| **change(1)** | Indicates that the packet may change en-route. See RFC 2460. |

## IPVersionTypeEnum Enumeration

The literals of the IPVersionTypeEnum enumeration are given in **Table 3‑87**.

Table ‑. 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. RFC 1347). |

## IANAHardwareTypeEnum Enumeration

The literals of the IANAHardwareTypeEnum enumeration are given in **Table 3‑88**.

Table ‑. 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. |

## IANAEtherTypeEnum Enumeration

The literals of the IANAEtherTypeEnum enumeration are given in **Table 3‑89**.

Table ‑. 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. |

## IANAAssignedIPNumbersTypeEnum Enumeration

The literals of the IANAAssignedIPNumbersTypeEnum enumeration are given in **Table 3‑90**.

Table ‑. 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 Control 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. |

## IANAPortNumberRegistryTypeEnum Enumeration

The literals of the IANAPortNumberRegistryTypeEnum enumeration are given in **Table 3‑91**.

Table ‑. 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. |

## MFlagTypeEnum Enumeration

The literals of the MFlagTypeEnum enumeration are given in **Table 3‑92**.

Table ‑. 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). |

# 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.

Acknowledgments

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

Participants:

Dean Thompson, Australia and New Zealand Banking Group (ANZ Bank)

Bret Jordan, Blue Coat Systems, Inc.

Adnan Baykal, Center for Internet Security (CIS)

Liron Schiff, Comilion (mobile) Ltd.

Jane Ginn, Cyber Threat Intelligence Network, Inc. (CTIN)

Richard Struse, DHS Office of Cybersecurity and Communications (CS&C)

Ryusuke Masuoka, Fujitsu Limited

Eric Burger, Georgetown University

Jason Keirstead, IBM

Paul Martini, iboss, Inc.

Jerome Athias, Individual

Sanjiv Kalkar, Individual

Terry MacDonald, Individual

Alex Pinto, Individual

Patrick Maroney, Integrated Networking Technologies, Inc.

Wouter Bolsterlee, Intelworks BV

Joep Gommers, Intelworks BV

Sergey Polzunov, Intelworks BV

Rutger Prins, Intelworks BV

Andrei Sîrghi, Intelworks BV

Jonathan Baker, MITRE Corporation

Sean Barnum, MITRE Corporation

Mark Davidson, MITRE Corporation

Ivan Kirillov, MITRE Corporation

John Wunder, MITRE Corporation

Mike Boyle, National Security Agency

Jessica Fitzgerald-McKay, National Security Agency

Takahiro Kakumaru, NEC Corporation

John-Mark Gurney, New Context Services, Inc.

Christian Hunt, New Context Services, Inc.

Andrew Storms, New Context Services, Inc.

Igor Baikalov, Securonix

Bernd Grobauer, Siemens AG

John Anderson, Soltra

Trey Darley, Soltra

Paul Dion, Soltra

Brandon Hanes, Soltra

Ali Khan, Soltra

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

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date** | **Editor** | **Changes Made** |
| wd01 | 08 December 2015 | Desiree Beck Trey Darley Ivan Kirillov Rich Piazza | Initial transfer to OASIS template |